RADC-TDR-64-310

COMPUTER PROGRAMMING TECHNIQUES FOR INTELLIGENCE ANALYST APPLICATION

TECHNICAL DOCUMENTARY REPORT NO. RADC-TDR-64-310

October 1964

Information Processing Branch

Rome Air Development Center
Research and Technology Division
Air Force Systems Command
Griffiss Air Force Base, New York

System 438L

(Prepared under Contract No. AF 30(602)-3303 by Thomas J. Watson Center, Yorktown Heights, New York.)

Quarterly Report No. 2.

KEYWORD LIST: Intelligence; Digital Computer, Statistics

ABSTRACT: This report presents a statement of progress on work supported by contract AF 30(602)-3303, during the period February 16, 1964, to May 15, 1964. Reports on computer program development, experimental studies, and designs of new facilities are included, for each of the five tasks specified in the work statement of the contract.

PUBLICATION REVIEW

This report has been reviewed and is approved. For further technical information on this project, contact

Approved: FRANK J. TOMAINI
Chief, Info Processing Branch
Intel & Info Processing Division

Approved: ROBERT J. QUINN, JR., Colonel, USAF
Chief, Intel & Info Processing Division

AIR FORCE, Cato Show Prtg Co 11 24 64 - 120

COMPUTER PROGRAMMING TECHNIQUES
FOR INTELLIGENCE ANALYST APPLICATION

REPORT NO. 2

INTRODUCTION

During the period covered by this report extensive modifications were made on the monitor and application programs of the AN/GYA complex. The multi-processing monitor can handle dynamic tape allocation, and it permits updating of the drum-disk library. There is satisfactory work progress on the multi-processing link between FMS and the AN/GYA monitor.

Additional distribution functions have been programmed for inclusion in STORM and on the AN/GYA disk. Facilities have been extended, and several integrated experiments were performed. A list-processing language has been defined, and is being programmed, which will facilitate inclusion of prototype studies in the lay user's system. Description of such prototypes has been continued.

Programs have been developed to permit experimental runs for the determination of statistical word association. The clear text system has been augmented by novel encoding features which permit faster search. Pre-processing of an extended data base is under way. A study was prepared proposing the use of an adaptive thesaurus as an experimental tool to quantify properties of indexing systems.

Bibliography on man-machine consoles and studies of systems have been extended.

There has been progress on the programming work connected with the debugging system.

Many of the modifications and program extensions were tried out extensively, on the AN/GYA complex. The most conspicuous new feature was, during the period covered by this report, the on-line updating facility.

TASK I: INVESTIGATION OF STATISTICAL PREDICTION, DISCRIMINATION, AND CLASSIFICATION TECHNIQUES

1. Definition of List Processor Functions for Lay User's Language.

A series of operations has been defined, tentatively, which will facilitate adaptation of the lay user's language to the AN/GYA system. These operation codes are to be regarded as macrostatements which can be called and executed under EXST (executive routine for statistical language on console), this approach will greatly facilitate programming and addition of operations. The macros are simply stored, similar to large data matrices, on the standard BCD input tape (currently A5). Since EXST operates under the multiprogramming monitor, execution of the lay user's language will, just as STORM, take place simultaneously with standard operations on the IBM 7094.

DF, A, DISPLAY DISPLAY DISPLAY $ DISPLAY etc.

> A field (matrix) A contains a message ready for display on the console. A $ sign separates lines of display.

WC, A

> Write contents of A into console buffer (i.e., display if ready).

RC, B

> Similar to DEFIN in EXST. Read the typed message and enter into buffer B.

CI,. B, N,

> Similar to second phase of DEFIN, i.e., the BCD message in buffer B is translated into floating point numbers, stored in list N.

CE, B, N, F

> Similar to DISPL in EXST. Internal floating point list B is translated into BCD list N which has FORTRAN F format.

BM, Z (a, b, c, d, e) = A, B, C, D, E, F.

A statement may be preceded by a statement label (A, B, etc.). If the contents of word Z (a BCD word) is equal to a (spelled out), go to statement A. Similarly, if Z = b, go to B, etc. If Z is neither of the words in parentheses, go to F. (Restricted to at most five words.)

BR, A

Unconditional branch to A.

LT, A, N

Loop transfer. Execute statements up to A, N times.

RC, B
EQU, (K) = B
DF, A, NAME OTHER STOCKS RELATED TO (K)

The foregoing example illustrates a conditional DF. At execution time, the symbol (K) is replaced by the contents of B (in the example, a word which has been typed).

LIST, ((I)) = FIRST, SECOND, THIRD
LT, α, 3
DF, A, ENTER THE ((I)) ROW
. . .
α

This sequence illustrates a loop executed three times, with ((I)) replaced by FIRST, SECOND, and THIRD consecutively.

RT, A, N

Read a file from tape unit A into field N.

WT, A, N

Write a file from field N onto tape unit A.

MV, A, B

Move contents of A to B.

ET, NAME, R, C, A

Enter the matrix NAME with R rows and C columns into the symbol table A (there may be several such tables).

XT, A, F, K

Extract the information (location, equivalent, rows, columns) of matrix named F in symbol table A and store result into K.

IN, A, N, N1, B, M

From field A, move characters N to N1 into field B, starting with M.

SCS, A Set control source to A
SDS, A Set data source to A

These specify the unit (tape, console, card reader) in which source statements and data can be found. The option is necessary not only for dynamic tape allocation, but also to enable medium-sized matrices to be entered from card reader.

The foregoing operation codes may undergo modification in the course of programming and application.

2. Extension of Distribution Routines

Numerical analysis and programs of non-central distributions for the statistical language are presented in Appendix I.

3. Examples for Lay User's Language (continued from previous report)

Biological Assay

The typical experimental situation is as follows. Several (N) groups of subjects (insects, people, etc.) are available. The group sizes are usually different. Groups are subjected to increasing "doses" of one or more treatments. A growth curve is to be fitted to the proportions affected under each dose. Most frequent applications are insecticide studies and learning experiments. The program performs a logistic transformation and estimates the growth curve by maximum likelihood.

Frame 1: NAME YOUR INDEPENDENT VARIABLES, SEPARATED BY COMMAS IF MORE THAN ONE. HOW MANY ARE THERE? HOW MANY GROUPS OF OBJECTS DO YOU HAVE?

Action 1: If names are NAME 1, NAME 2, enter these into lists (L1), (L2), etc., to be inserted below. Enter number of variables into (M); number of groups into (N).

Frame 2: NAME YOUR DEPENDENT VARIABLE.

Action 2: Enter into list (LD1)

Repeat through Action 4 (M) times with (L) taking the value (L1), (L2), ... (LM).

Frame 3: ENTER DATA FOR (L) IN THE FORM A1, A2, A3, ETC., WHERE A1, A2, A3 ARE ACTUAL NUMBERS WITH OR WITHOUT DECIMAL POINTS.

Action 3: Transfer to EXST-routine DEFIN, set up the contents of (L1), (L2), etc. (i.e. the names) into dictionary table. Compare each number of entries with (N). If discrepancies occur, display:

ERROR IN ENTRY. REPEAT FROM THE FOLLOWING POINT:

then go back to the end of Action 2, i.e., entry point of the loop.

Frame 4: DO YOU WISH TO TRANSFORM DATA POINTS OF (L) INTO LOGS, EXP, SQU, OR ROOT?

Action 4: If answer is no, proceed to loop entry. If answer is any one of the four functions, go to corresponding STORM function under EXST, using the same argument for input and output, e.g., ROCT, NAME 1, NAME 1. Store transformation names into (T1), (T2), etc. If no transformation, store blank at that place.

Needed revision of EXST: Include LOG and EXP routines on disk.

END OF LOOP

Action 5: When loop is completed, execute HPAC, NAME 1, NAME NAMEM = X.

Frame 6: ENTER THE TOTAL NUMBER OF OBJECTS (INSECTS, EMPLOYEES, ETC.) IN EACH GROUP (UNDER EACH DOSE OR TREATMENT).

Action 6: Check that number of entries is (N). If not, display ERROR and re-display frame 6. Then enter via DEFIN, *TOT*, N, 1.

Frame 7: ENTER THE NUMBER OF SUCCESSES (INSECTS KILLED, EMPLOYEES CURED, ETC.) UNDER EACH DOSE OR TREATMENT, IN THE SAME MANNER.

Action 7: Test N. If incorrect, display ERROR, REPEAT and redisplay frame 7.

Execution:

```
ONEX, N, 1 = ON
DUP, PO, PE
DUP, *TOT*, W
SUB, ON, PE, QE
DIV, PE, QE, R
LOG, R, LR
SCM, LR, 0.5, YPR
DUP, YPR, Y
```

Then repeat, ten times, the following ten statements (or macros).

```
WREG, X, Y, N, W, B, YPR
LGIT, YPR, N, PE
SUB, PO, PE, DIFF
SUB, ON, PE, QE
MPY, PE, QE, PQ
SCM, PQ, 2, DEN
DIV, DIFF, DEN, COR
ADD, YPR, COR, Y
SCM, DEN, 2, PR
MULT, *TOT*, PR, W
```

The macro-routines are as follows:

```
   LGIT, A, N, B
   SCM, A, 2, A2
   ONEX, N, 1, O
   EXP, A2, NUM
   ADD, O, NUM, DEN
   DIV, NUM, DEN, B
   ENDMAC
```

and

```
   WREG, X, Y, N, W, B, YPR
   WROW, X, W, XW
   WROW, Y, W, YW
   ONEX, N, 1, ON
   HPAC, W, XW, XX
   HPAC, ON, X, XZ
   MPYT, XZ, XX, S
   INV, S, SIN
   MPYT, XZ, YW, RHS
   MPY, SIN, RHS, B
   MPY, XZ, B, YPR
   ENDMAC
```

Note: Include in EXST, the following routines:

DUP, A, B	duplicate
WROW, A, B, C	weigh rows
WCOL, A, B, C	weigh columns

A is a matrix. In WROW, B is a column vector whose first element is multiplied into the first row of A, second into second row of A, etc. Thus, if the elements of B were the diagonal elements of a diagonal matrix D, C = DA. In WCOL, B is a row vector whose first element is multiplied into the first column of A, second into second column of A, etc. Thus, if the elements of B were the diagonal elements of a diagonal matrix D, C = AD.

After execution, state:

Frame 8: THE FOLLOWING DISPLAY IS THE REGRESSION EQUATION. IT HAS ONE COLUMN. THE FIRST ELEMENT IS THE CONSTANT TERM. THE OTHER TERMS ARE THE COEFFICIENTS OF:

(T1) (L1)
(T2) (L2)
(T3) (L3)
 ⋮ ⋮
(TM) (LM)

Comment: The latter are inserts from Actions 1 and 4.

Action 8: When user says GO, execute DISPL, B.

Frame 9: THE FOLLOWING DISPLAY SHOWS THE OBSERVED PROPORTIONS OF SUCCESS IN THE FIRST COLUMN, AND THE EXPECTED PROPORTIONS IN THE SECOND COLUMN.

Action 9: When user says GO, execute DISPL, PO, PE, RESULT.

Frame 10: THIS IS THE END. IF YOU WISH TO START A NEW JOB, TYPE CANCEL. IF YOU WISH TO SIGN OFF, TYPE KAPUT. THANK YOU.

Frame 1: THIS PROGRAM WILL SET UP A SAMPLING INSPECTION PLAN FOR ACCEPTING OR REJECTING A LOT OF ARTICLES SUBMITTED FOR INSPECTION.

ON EXAMINING AN ARTICLE OF THE LOT SUBMITTED FOR INSPECTION, CAN YOU IDENTIFY IT AS DEFECTIVE OR NON-DEFECTIVE?

Action 1: The answer can be yes or no. If the answer is no, then display THIS PROGRAM CAN BE USED ONLY WHEN AN ARTICLE CAN BE IDENTIFIED AS DEFECTIVE OR NON-DEFECTIVE. Then skip to EXIT. If the answer is yes, go to Frame 2.

Frame 2: WHICH SAMPLING PROCEDURE DO YOU WISH TO ADOPT?

1 ONE SAMPLE
2 TWO SAMPLES
3 A SEQUENCE OF SAMPLES ON ONE ARTICLE DRAWN ONE AT A TIME

TYPE THE NUMBER PRECEDING THE DESIRED METHOD.

Action 2: The answer can be 1, 2, or 3. If the answer is 1, go to Frame 3. If the answer is 2, go to Frame 11. If the answer is 3, go to Frame 23.

Frame 3: DO YOU WANT TO STATE THE SIZE OF THE SAMPLE?

Action 3: The answer can be yes or no. If the answer is yes, go to Frame 4. If the answer is no, go to Frame 10.

Frame 4: STATE YOUR SAMPLE SIZE.

Action 4: The answer will be a number. Equate if to N internally.

Frame 5: IF YOU CONSIDER CONSUMERS RISK THE IMPORTANT ONE, TYPE LETTER A. IF YOU CONSIDER PRODUCERS RISK IMPORTANT, TYPE LETTER B.

Action 5: The answer can be A or B. If the answer is A, go to Frame 6. If the answer is B, go to Frame 8.

Frame 6: STATE CONSUMERS RISK AS A PROPORTION. IF YOU DO NOT KNOW, WE RECOMMEND YOU STATE 0.05.

Action 6: The answer will be a number. This number will internally be called CR. Go to Frame 7.

Frame 7: STATE THE PROPORTION OF DEFECTIVES, WHICH THE CONSUMER WILL TOLERATE.

Action 7: The answer will be a number. Call this number PT.

Execution:

BINOP, CR, N, PT = CC

Then display:

Display 1: THE SAMPLING SCHEME IS AS FOLLOWS:

DRAW A RANDOM SAMPLE OF SIZE (N) AND DETERMINE THE NUMBER OF DEFECTIVES. IF THE NUMBER OF DEFECTIVES IS GREATER THAN (CC), REJECT, OTHERWISE ACCEPT, THE WHOLE LOT.

Comment: (N) and (CC) will be the numbers obtained in queries and execution.

END OF JOB

Frame 8: STATE PRODUCERS RISK AS A PROPORTION. IF YOU DO NOT KNOW, WE RECOMMEND YOU STATE 0.10.

Action 8: The answer will be a number. Call this number PR. Then go to Frame 9.

Frame 9: STATE THE PROPORTION OF DEFECTIVES CLAIMED BY THE PRODUCER.

Action 9: The answer will be a number. Call it PP.

Programming instructions: The program will then call:

BINOP, PR, N, PP = CC

then go to Display 1.

Frame 10: SUPPLY THE FOLLOWING INFORMATION, NUMBERS SEPARATED BY COMMAS:
CONSUMERS RISK,
(IF UNKNOWN, WE RECOMMEND 0.05)
PRODUCERS RISK,
(IF UNKNOWN, WE RECOMMEND 0.10)
PROPORTION DEFECTIVE WHICH CONSUMER WILL TOLERATE,
PROPORTION DEFECTIVE CLAIMED BY PRODUCER

Action 10: The answer will be four numbers. These numbers will be stored as follows: CR, PR will be a 2x1 matrix called R. PT and PP will be a 2x1 matrix called P.
A subroutine will be prepared:
SISIP, R, P = C
This program will compute the required sample size (N) which is the first element of C, and the action point (CC) which is the second element of C. These will be included in Display 1.
GO TO DISPLAY 1

Frame 11: DO YOU WANT TO SPECIFY THE SAMPLE SIZES?

Action 11: The answer can be yes or no. If the answer is yes, go to Frame 12. If the answer is no, go to Frame 14.

Frame 12: SPECIFY THE SIZE OF THE FIRST SAMPLE.

Action 12: The answer will be a number. Call it N1. Then go to Frame 13.

Frame 13: SPECIFY THE SIZE OF THE SECOND SAMPLE.

Action 13: The answer will be a number. Call it N2. Then go to Frame 19.

Frame 14: DO YOU WANT TO PUT A LIMIT ON THE MAXIMUM OR MINIMUM OF THE TOTAL SAMPLE SIZE?

Action 14: The answer can be yes or no. If the answer is yes go to Frame 15. If the answer is no go to Frame 19.

Frame 15: DO YOU WANT TO SPECIFY THE MAXIMUM TOTAL SAMPLE SIZE?

Action 15: Set a word named INDEX = 0. The answer can be yes or no. If the answer is no go to Frame 17 otherwise go to Frame 16.

Frame 16: SPECIFY THE MAXIMUM TOTAL SAMPLE SIZE.

Action 16: The answer will be a number. Call the number N. Put INDEX = 1, and go to Frame 17.

Frame 17: DO YOU WANT TO SPECIFY THE MINIMUM TOTAL SAMPLE SIZE?

Action 17: The answer can be yes or no. If the answer is no then go to Execution I1. Otherwise, go to Frame 18.

Frame 18: SPECIFY THE MINIMUM TOTAL SAMPLE SIZE.

Action 18: The answer will be a number. Call the number N1. Put INDEX = INDEX + 1. Go to Execution I1.

<u>Execution I1:</u>

If INDEX = 1, put N1 = N2 = N/2.
If INDEX = 2, put N2 = N - N1.

Go to Frame 22.

Frame 19: CAN YOU SPECIFY THE SIZE OF THE LOT?

Action 19: The answer can be yes or no. If the answer is no go to Frame 21, otherwise go to Frame 20.

Frame 20: SPECIFY THE SIZE OF THE LOT.

Action 20: The answer will be a number. Call it NN. Then calculate N = NN/20. and N/ = N2 = N/2. Then go to Frame 22.

Frame 21: IS IT ALL RIGHT IF BOTH THE SAMPLES ARE OF SIZE 50?

Action 21: The answer can be yes or no. If the answer is yes, take N1 = N2 = 50, and go to Frame 22. If the answer is no then display
THE PROGRAM CANNOT BE USED, and go to EXIT.

Execution I2
A double sampling inspection routine will be executed (program not yet available) which has output C1 and C2, to appear in the following display.
Go to D2

Display D2:

THE DOUBLE SAMPLING INSPECTION PLAN IS AS FOLLOWS.

DRAW A RANDOM SAMPLE OF SIZE N1 AND COUNT THE NUMBER OF DEFECTIVES.

IF THE NUMBER OF DEFECTIVES IS LESS THAN C1, ACCEPT THE WHOLE LOT.

IF THE NUMBER OF DEFECTIVES IS GREATER THAN C2, REJECT THE WHOLE LOT.

OTHERWISE DRAW ANOTHER SAMPLE OF SIZE N2 AND COUNT THE NUMBER OF DEFECTIVES.

IF THE COMBINED NUMBER OF DEFECTIVES IS LESS THAN C2, ACCEPT; OTHERWISE REJECT THE WHOLE LOT.

END OF JOB.

Frame 22: Same as Frame 10.

Action 22: The answer will be four numbers. These numbers will be called CR, PR, PT, PP respectively. Proceed to Execution I2.

Frame 23: Same as Frame 10.

Action 23: The answer will be four numbers. These numbers will be denoted internally as CR, PR, PT, PP. Then proceed to Execution I3.

Execution I 3.

```
AL = 1.-CR
BE = 1.-PR
PPC = 1.-PP
AA = LOGF (BE/CR) / (LOGF(PP/PT)
    - LOGF (PPC/PTC))
BB = LOGF (PTC/PPC) / (LOGF(PP/PT)
    - LOGF (PPC/PTC))
CC = LOGF (PR/AL) / (LOGF(PP/PT)
    - LOGF (PPC/PTC))
        Go to Display D3
```

Display D3: FOR YOUR INSPECTION PLAN, YOU WILL NEED TWO NUMBERS AS YOU DRAW EACH SAMPLE:

AN = (AA) + (BB) TIMES (NUMBER OF SAMPLES)
BN = (CC) + (BB) TIMES (NUMBER OF SAMPLES)

IF YOU WISH TO SEE THE NUMERICAL VALUES OF AN AND BN FOR N=1, 50, TYPE YES, OTHERWISE TYPE NO.

Action: If YES is typed, then display D4. If the answer is NO, then display D5.

Display D4: THE DISPLAY WHICH FOLLOWS HAS THREE COLUMNS. THE FIRST COLUMN INDICATES THE NUMBER OF SAMPLES DRAWN. THE SECOND COLUMN IS A NUMBER BN, THE THIRD COLUMN IS A NUMBER AN.

Action: Display a matrix:

Column 1: N (values 1 to 50)
Column 2: BN = AA + BB*N
Column 3: AN = CC + BB*N

Display D5: THE SEQUENTIAL SAMPLING INSPECTION PLAN WILL BE AS FOLLOWS:

SAMPLES WILL BE DRAWN SEQUENTIALLY ONE AT A TIME AND EACH TIME THE TOTAL NUMBER OF DEFECTIVES WILL BE COMPARED WITH AN AND BN.

IF THE NUMBER OF DEFECTIVES AMOUNG N SAMPLES, DN, IS LESS THAN OR EQUAL TO BN, THEN ACCEPT THE LOT.

IF DN IS GREATER THAN OR EQUAL TO AN, THEN REJECT THE LOT.

IF DN IS BETWEEN BN AND AN, THEN DRAW ANOTHER SAMPLE AND REPEAT THE PROCESS.

Action: **Then call EXIT.**

System Reliability Analysis

The following is a description of a procedure for evaluating the reliability of a system consisting of a set of subsystems in series with each subsystem a set of components in parallel. The user need only specify the number of subsystems, the number of components in each subsystem, and the mean time to failure for each component. He may then obtain the probability that the system survives to any specified time.

Display: THIS PROGRAM PERMITS AN ANALYST TO OBTAIN THE PROBABILITY OF SYSTEM SURVIVAL TO ANY ARBITRARY TIME FROM KNOWLEDGE OF THE MEAN TIME TO FAILURE FOR EACH COMPONENT.

THE SYSTEM IS A SET OF SUBSYSTEMS IN SERIES FOR WHICH EACH SUBSYSTEM IS A SET OF COMPONENTS IN PARALLEL. THAT IS, THE SYSTEM FAILS WHEN ANY SUBSYSTEM FAILS AND A SUBSYSTEM FAILS ONLY WHEN ALL ITS COMPONENTS FAIL.

Frame 1: ARE YOU WILLING TO ASSUME AN EXPONENTIAL RELIABILITY FUNCTION FOR EACH COMPONENT?

NOTE: THE EXPONENTIAL RELIABILITY FUNCTION IS OFTEN AT LEAST A GOOD APPROXIMATION TO THE TRUE RELIABILITY FUNCTION. IT IS THE APPROPRIATE RELIABILITY FUNCTION FOR ANY COMPONENT FOR WHICH FAILURE IS PRIMARILY DUE TO CHANGE RATHER THAN WEAR.

Action 1: The answer can be yes or no. If the answer is yes, to to Frame 3. If the answer is no, go to Frame 2.

Frame 2: AT THIS STAGE, THE PRESENT PROGRAM HAS NOT INCORPORATED THE OPTION OF RELIABILITY FUNCTIONS OTHER THAN THE EXPONENTIAL.

IF YOU REQUIRE THE USE OF A DIFFERENT RELIABILITY FUNCTION, YOU SHOULD PERFORM THE ANALYSIS BY ANOTHER METHOD.

WOULD YOU LIKE TO ASSUME AN EXPONENTIAL RELIABILITY FUNCTION FOR EACH COMPONENT AND USE THE RESULT AS A FIRST APPROXIMATION?

Action 2: If the answer is yes, go to Frame 3. If no, go to PROGRAM TERMINATED.

Frame 3: DOES THE SYSTEM CONSIST OF SUBSYSTEMS WHERE THE SUBSYSTEMS ARE IN SERIES AND THE COMPONENTS OF EACH SUBSYSTEM ARE IN PARALLEL? (NOTE: IT IS NOT NECESSARY THAT COMPONENTS IN A GIVEN SUBSYSTEM BE IDENTICAL).

NOTE: THE SUBSYSTEMS ARE IN SERIES IF THE SYSTEM FAILS WHEN ANY ONE SUBSYSTEM FAILS. A SUBSYSTEM HAS ITS COMPONENTS IN PARALLEL IF THE SUBSYSTEM FAILS ONLY WHEN ALL ITS COMPONENTS HAVE FAILED.

Action 3: If the answer is yes, go to Frame 5; if no, go to Frame 4.

Frame 4: THE PRESENT PROGRAM IS NOT DESIGNED TO HANDLE YOUR SYSTEM. IT MAY BE POSSIBLE TO BREAK YOUR SYSTEM DOWN INTO SUBSYSTEMS, EACH OF WHICH IS OF A FORM REQUIRED FOR THIS PROGRAM.

DO YOU WISH TO START OVER?

Action 4: If the answer is yes, go back to Frame 1. If no, display: SORRY, AT THIS STAGE, THE PRESENT PROGRAM IS NOT DESIGNED TO HANDLE YOUR SYSTEM. PROGRAM TERMINATED. GO TO PROGRAM TERMINATED.

Frame 5: SPECIFY THE NUMBER OF COMPONENTS IN EACH SUBSYSTEM IN ORDER. FOR EXAMPLE, 3, 4, 7, 2, 2, INDICATES A SYSTEM OF FIVE SUBSYSTEMS, THE

FIRST OF WHICH CONTAINS 3 COMPONENTS, THE SECOND 4 COMPONENTS, THE THIRD 7 COMPONENTS, ETC.

NOTE: YOU ARE LIMITED TO A MAXIMUM OF 20 SUBSYSTEMS AND EACH SUBSYSTEM IS LIMITED TO 30 COMPONENTS.

Action 5: The answer will be a vector of positive elements (with no more than 20 elements each of which is a positive integer less than 31). Call this vector P.

Frame 6: ENTER DATA FOR FIRST SUBSYSTEM. YOU SHOULD ENTER THE MEAN TIME TO FAILURE FOR EACH COMPONENT IN THIS SUBSYSTEM. THEY MAY BE IN ANY ORDER.

Action 6: The answer will be a vector of nonnegative numbers, the total number of which is identical with the first element of the vector P.

Frame 7: ENTER DATA FOR NEXT SUBSYSTEM. YOU SHOULD ENTER THE MEAN TIME TO FAILURE FOR EACH COMPONENT IN THIS SUBSYSTEM. THEY MAY BE IN ANY ORDER.

Action 7: A vector of nonnegative numbers should be entered. The size of this vector should be equal to the element in the vector P corresponding to this subsystem.

After the data is entered, if there are more subsystems, data to be entered (as indicated by the number of elements in P) repeat frame 7 (this is a loop). If data is sufficient, as indicated by the vector P, display DATA COMPLETE and go to Frame 8.

Frame 8: YOU MAY NOW EVALUATE THE PROBABILITY OF SYSTEM SURVIVAL TO ANY SPECIFIED TIME. ENTER THE TIME POINTS FOR WHICH YOU REQUIRE THIS PROBABILITY. YOU ARE LIMITED TO 50 TIME POINTS AND THEY MAY BE IN ANY ORDER. THE OUTPUT WILL INCLUDE YOUR TIME POINTS. TYPE

"EVAL" WHEN YOU ARE FINISHED WITH YOUR DATA INPUT.

Action 8: The answer will be a vector of positive values with a maximum of 50 elements. Call this vector T. When EVAL is typed, perform the operation REVAL P, T, M=R (defined at the end of this write-up) where this operation evaluates the system reliability function for each value in the vector T. The output vector R is the required set of reliabilities. P has been previously defined. The vectors T and R are displayed side by side. Go to Frame 9.

Frame 9: DO YOU WISH TO EVALUATE THE PROBABILITY OF SYSTEM SURVIVAL FOR ADDITIONAL TIME POINTS?

Action 9: If the answer is yes, go to Frame 8; if no, go to Frame 10.

Frame 10: WOULD YOU LIKE TO CONSIDER CHANGING YOUR SYSTEM FOR PURPOSES OF COMPARISON? FOR EXAMPLE, YOU MAY WISH TO CONSIDER THE EFFECTS OF INCREASING THE NUMBER OF COMPONENTS IN ONE OR MORE SUBSYSTEMS TO INVESTIGATE THE IMPROVEMENT IN SYSTEM RELIABILITY.

Action 10: If the answer is yes, go to Frame 11; if no, go to PROGRAM TERMINATED.

Frame 11: DO YOU WISH TO CHANGE OR DELETE ANY OF THE PREVIOUSLY SPECIFIED SUBSYSTEMS?

Action 11: If the answer is yes, go to Frame 12; if no, go to Frame 15.

Frame 12: ENTER A 1 FOR EACH SUBSYSTEM YOU WISH TO CHANGE, A 0 FOR EACH SUBSYSTEM TO BE LEFT UNCHANGED, AND A 2 FOR EACH SUBSYSTEM TO BE DELETED. (YOU SHOULD MAINTAIN THE SAME ORDER AS USED PREVIOUSLY FOR THE SUBSYSTEMS).

Action 12: The data for subsystems correcponding to each 2 should be deleted. The data for each subsystem corresponding to a 0 should be retained. The data for each subsystem corresponding to a 1 should be displayed for change, one subsystem at a time. Upon display, any part of the subsystem data may be "erased" and replaced by new data i.e., if there are any changes in a subsystem (if at least one 1 is entered for Frame 12) then display the original data for the first subsystem to be changed, head it with the statement of Frame 13 and go to Frame 13. If these are only deletions and no changes, go to Frame 15.

Frame 13: DO YOU WISH TO ERASE ANY DATA IN THE FOLLOWING SUBSYSTEM? YOU SHOULD ERASE ANY DATA YOU WISH TO ELIMINATE OR CHANGE.

Action 13: If yes, retain data display but replace heading with statement of Frame 14 and go to Frame 14. If no, retain data display under the heading of Frame 15, and go to Frame 15.

Frame 14: ENTER THE DATA TO BE ERASED. THE ENTRIES MAY BE IN ANY ORDER BUT IF THE SAME NUMBER IS TO BE ERASED MORE THAN ONCE, YOU MUST ENTER IT AS MANY TIMES AS IT IS REQUIRED TO BE ERASED.

Action 14: Positive numbers identical to some of the subsystem data are entered. All corresponding elements of the subsystem data are deleted, one for each erasure entry. The corrected data is displayed under the heading CORRECTED DATA. When GO is typed, the corrected data is retained in display with the statement of Frame 15 in the heading. Go to Frame 15.

Frame 15: DO YOU WISH TO ADD COMPONENTS TO THIS SUBSYSTEM?

Action 15: If yes, go to Frame 16. If no and there are no additional subsystems to be changed, go to Frame 17, otherwise insert data for next subsystem to be changed into Frame 13 and go to Frame 13. This is a loop.

Frame 16: ENTER THE MEAN TIME TO FAILURE FOR EACH NEW COMPONENT AND TYPE ADD WHEN YOU ARE FINISHED ENTERING NEW DATA. NOTE; THE TOTAL NUMBER OF COMPONENTS IN A SUBSYSTEM MAY NOT EXCEED 30.

Action 16: A vector of positive numbers is entered. The total number of these elements plus the total number of subsystem data left after erasure should not exceed 30. When ADD is typed, the entered data should be included with the corrected subsystem data and displayed under the heading COMPONENT MEAN TIMES TO FAILURE FOR NEW SUBSYSTEM. When GO is typed, if no more subsystems are to be changed, go to Frame 17, otherwise insert data for next subsystem to be changed under the statement of Frame 13 and go to Frame 13.

Frame 17: DO YOU WISH TO ADD MORE SUBSYSTEMS TO THE SYSTEM?

Action 17: If yes, go to Frame 18; if no, go to Frame 21.

Frame 18: SPECIFY IN ANY ORDER THE NUMBER OF COMPONENTS IN EACH SUBSTEM TO BE ADDED TO THE SYSTEM. FOR EXAMPLE, 2, 5 INDICATES THAT YOU REQUIRE TWO ADDITIONAL SUBSYSTEMS WITH TWO COMPONENTS IN THE FIRST SUBSYSTEM AND FIVE IN THE SECOND.

NOTE: THE TOTAL NUMBER OF SUBSYSTEMS MAY NOT EXCEED 20.

Action 18: The answer will be a vector of positive integers.

Frame 19: ENTER DATA FOR FIRST SUBSTEM TO BE ADDED. YOU SHOULD ENTER THE MEAN TIME TO FAILURE FOR EACH COMPONENT IN THIS SUBSYSTEM. THEY

MAY BE IN ANY ORDER.

NOTE: THE TOTAL NUMBER OF COMPONENTS IN A SUBSYSTEM MAY NOT EXCEED 30.

Action 19: The answer will be a vector of nonnegative numbers, the total number of which is identical with the corresponding element of PP. If there are more subsystems to be added, go to Frame 20, otherwise go to Frame 21.

Frame 20: ENTER DATA FOR NEXT SUBSYSTEM TO BE ADDED. YOU SHOULD ENTER THE MEAN TIME TO FAILURE FOR EACH COMPONENT IN THIS SUBSYSTEM. THEY MAY BE IN ANY ORDER.

Action 20: The answer should be a vector of nonnegative numbers the total number of which must be equal to the corresponding element of PP. If there are more subsystems to be added, go to Frame 20 (this is a loop), otherwise go to Frame 21.

Frame 21: DO YOU WISH TO SEE THE COMPONENT DATA FOR THE ENTIRE NEW SYSTEM?

Action 21: All changed, unchanged, and new subsystem data are organized into the proper form for a new system. A new vector PPP is defined for which the i^{th} element indicates the number of components in the new i^{th} subsystem. The vector PPP defines the form of the new system.

If no, go to Frame 22. If yes, display data for entire system with each subsystem numbered. Go to Frame 22.

Frame 22: YOU MAY NOW EVALUATE THE PROBABILITY THAT THE NEW SYSTEM WILL SURVIVE TO ANY SPECIFIED TIME. ENTER THE TIME POINTS FOR WHICH YOU REQUIRE THIS PROBABILITY. YOU ARE LIMITED TO 50 TIME POINTS AND THEY MAY BE IN ANY ORDER. THE OUTPUT WILL INCLUDE YOUR TIME POINTS. TYPE EVAL WHEN YOU ARE FINISHED WITH YOUR DATA INPUT.

Action 22: The answer will be a vector of positive values with a maximum of 50 elements. Call this vector TT. When EVAL is typed, perform the operation REVAL PPP, TT = RR, where this operation evaluates the system reliability function for each value in the vector TT. The output vector RR is the required set of reliabilities. PP has been previously defined. Display TT and RR side by side. Go to Frame 23.

Frame 23: DO YOU WISH TO EVALUATE THE PROBABILITY OF SYSTEM SURVIVAL FOR ADDITIONAL TIME POINTS?

Action 23: If yes, go to Frame 22; if no, go to Frame 24.

Frame 24: DO YOU WISH TO CONSIDER ANOTHER CHANGE IN YOUR SYSTEM FOR PURPOSES OF COMPARISON?

Action 24: If yes, go to Frame 11 and use new system data. If no, go to PROGRAM TERMINATED.

The following is the FORTRAN program for evaluating the system reliability program. The subroutine is identified by:

REVAL P, T, M = R

Where P is the vector which defines the form of the system, T is the vector of time points, M is the vector of mean times to failure for every component in the system (arranged so that the components for the first subsystem are first, for the second subsystem are second, etc), and R is the output vector of reliabilities. The FORTRAN program is :

```
      SUBROUTINE REVAL (JP, T, M, K, N, R)
      DIMENSION JP(1), T(1), M(1), R(1)
      DO 20 I=1, N
      RI = 1.
      U = 1
      DO 21 J=1, K
      RIJ = 1.0
      U1 = U + JP(J) - 1
      DO 22 L=U, U1
22    RIJ = RIJ* (1.0-EXPF(-T(I)/M(L)))
      U = U1 + 1
      RIJ = 1.0-RIJ
21    RI = RI*RIJ
20    R(I) = RI
      RETURN
      END (1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0)
```

TASK II INVESTIGATION OF INTEGRATED COMPUTER ORIENTED INFORMATION RETRIEVAL TECHNIQUES

This task is concerned with a study of information retrieval techniques, and the development and expansion of computer programs to aid in applications.

2.1 Statistical Word Association

The edited input text, i.e., with common words removed and the remaining words normalized to canonical form, was run against the completed word pair generator program. This 7040 program produces, for each sentence in the text, a list of all the word pairs in that sentence. Word pairs are not generated by order but simply by occurrence. That is, if words A and B occur in the sentence, only one of the pairs AB and BA is produced. Since each word is replaced by its dictionary number, and since the number of dictionary entries for the text is approximately 15,000, one machine word is sufficient for each word pair. The sentence word pairs are serially added to the list of word pairs for the text. There were approximately 2.8 million word pairs in the sample. These pairs were sorted by a standard sort program. After sorting, another 7040 program combined equal word pairs and produced, for each non-unique pair, an entry containing the pair, the frequency of each word of the pair, the pair frequency, and the expected pair frequency under the assumption that their co-occurrence in sentences was independent. A 7090 program has been completed which will eliminate those pairs for which the observed frequency has a probability greater than .01 of occurring, under the assumption that the data follows a Poisson law. The calculation of probabilities is done with a routine taken from the STORM package. It is estimated that the run will take approximately four hours.

We now propose the development of a look-up program for the word pairs which remain after the 7090 elimination run. Present plans are to use the 7040-1301 system in order to reduce look-up time.

2.2 Adaptive Thesaurus

The first part of Appendix II contains a detailed description of properties and functions of indexing systems. This leads to a statement of motivation for an adaptive thesaurus. To fit this approach into a general framework, various kinds of indexing languages are described. A thesaurus is then considered as an explicit semantic dictionary in such a language. The paper is tutorial and contains

a) a definition of indexing;

b) the labeling process;

c) the dependence of content on the intent of the user;

d) the problems of keywords which have to reflect potential future requests;

e) the two types of translation required (document content into indexing language and query into indexing language);

f) the types of grammars (association with direction, proximity, etc);

g) question of semantics (canonical form of index terms, thesauri);

h) differences between machine translation and information retrieval.

The second part of Appendix II contains a discussion of some quantitative parameters for the evaluation of the efficiency and cost of indexing. On the basis of such parameters, some of the problems of designing an indexing system are discussed in qualitative terms. Quantification of such qualitative properties is suggested utilizing experimental results to be obtained with the aid of an adaptive thesaurus.

2.3 English Text Query System

2.3.1 Modification of Encoding

One modification has been made to the system to alter the method of storing data. This results in a large reduction in storage requirements and a substantial increase in operating speed.

A second modification to the query processor is being made to permit more general and versatile queries.

The new data storage method is a second data compression scheme to be incorporated in the system. To review very quickly the first scheme consisted of developing a word list of all distinct words in the data file, together with their frequency of occurrence, and assigning a code number to each word in the list, using short code numbers for the most frequently occurring words. The code number was then substituted for each English word in the text, resulting in a large reduction in the amount of storage required for the encoded text. For the text being used - the claims portion of a U.S. patent file - the reduction was about four-fold.

The new data compression technique is being used in the dictionary ("word list") part of the file. In addition to containing the English word and the code number, the claim numbers of the claims containing the word are also stored with the word. In the past, this was only done if the word occurred in less than six claims. It would have been desirable to do this for all words, but the list of claim numbers was excessively long for many common words.

The compression techniques to alleviate this problem might best be illustrated by an example.

COMPOSITE	462,	463,	464,	465
	466,	469,	470,	473
	2456,	2457,	4428,	4431
	4432			

There are occurrences of the word "COMPOSITE" in 13 claims. Since claim numbers in the present file can go to about 18,000, a naive listing of numbers would require 5 decimal digits, or 15 binary digits for each claim number.

The encoding scheme takes advantage of the fact that these numbers tend to occur in "bursts," that is, there are often several claim numbers which are close together numerically, and more specially, several consecutive claim numbers.

The encoding procedure is:

1) Record the first claim number as an 18 bit quantity.

2) Take the difference between adjacent claim numbers in the list.

3) If there is more than one consecutive difference of 1 (i.e., two or more consecutive claim numbers), record a 6 bit code character 75 (octal), followed by the number (+1) of consecutive claim numbers (assumed for the moment to be less that 61).

4) If step 3 does not apply but the difference is less than 61 (75 Octal) record the difference as a 6 bit quantity.

5) If the difference is greater than 61, but less than or equal to 4095, (7777 Octal) record a 6 bit code 76 (Octal), followed by the difference as a 12-bit quantity.

6) If the difference is greater than 7777 (Octal) record a 6-bit code of 77 (Octal) followed by the difference, as an 18-bit quantity.

The choice of six bits as a unit of length for code words was partly a matter of convenience and partly of experiment. Some further attention will be given to looking for a rule for the determination of the encoding rules for more general files.

The resulting calculations for the examples are shown in the table below.

Claim Number		Difference	Encoding
Decimal	Octal	Octal	Octal
462	716	--	00 07 16
463	717	1	75 07
464	720	1	- - - -
465	721	1	- - - -
466	722	1	- - - -
469	723	1	- - - -
470	724	1	- - - -
473	731	5	05
2456	4630	3677	76 36 77
2457	4631	1	01
4428	10514	3663	76 36 63
4431	10517	3	03
4432	10520	1	01

The compressed code requires 90 bits, compared to the original requirement of 13 x 15 - 195 bits. Data is still being collected on the exact savings possible with methods of this type.

2.3.2 Extension of Query

The query processor is being modified to accept more general query specifications. As first developed, the processor would accept a number, N, of words or phrases as "acceptance words," and any number of "rejection" words. A search could be made on a combination of two criteria:

1) find all claims containing at least M out of the N phrases;

2) but discard any claim that contains any "rejection" word.

For example, four relevant words might be "magnetic," "storage," "core," "memory," and retrievals might be based on finding at least any three of the four words.

However, a search would probably be more accurate if it were based on finding

one of the two words "magnetic," "core"

and one of the two words "memory," "storage."

This request was not possible with the original system, but will be easily handled in the modified version.

The modified system will accept and interpret queries according to the following simple rules:

1) Typing in a word or phrase (i.e., any string of alpha-numeric characters) will establish that phrase as a criterion for retrieval.

2) Depressing a "combination" key (one of the CCC Process Keys) will combine the preceding two criteria according to the appropriate rules and establish it as a new criterion, replacing the previous two criteria.

3) A search can be made whenever only one criterion remains.

For example:

Keyboard Input	Means that any one claim should be retrieved which contains:
Type "Memory"	the word "memory"
Type "Storage"	the word "storage"
Depress OR	either the word "storage" or "memory" or both
Type "Magnetic"	the word "magnetic"
Type "Tape"	the word "tape"
Depress AND NOT	the word "magnetic" but not "tape"
Magnetic	the word "magnetic"
Drum	the word "drum"
AND NOT	the word "magnetic" but not "tape"

OR "magnetic" but not "tape" or magnetic but not "drum"

which is the same as "magnetic" but not ("drum" or "tape")

AND ("memory" or "storage") and (magnetic but not ("drum" or "tape"))

If the "Search" key is now depressed, the search will be made on the basis of the last criterion.

2.3.3 Application of clear text system to formatted files

It became apparent recently that the present Unformatted file system will work very effectively as a query system for a formatted file. We are considering creating a test file to verify the ease of use.

2.3.4 Processing of data

An enlarged file of patent claims has been reprocessed to create a disk with the new compressed format. The new disk will be tested in the near future.

TASK III INVESTIGATION OF MULTI-PROCESSING TECHNIQUES FOR INTELLIGENCE INFORMATION PROCESSING

This task is concerned with the extension of operational multi-programming systems, and the development of procedures for multiple-console applications.

During the period covered by this report, program processing, loading, and testing as an integrated operation was successfully performed under control of the AN/GYA multiprocessor. This feature permitted the updating of the library of the statistical language during AN/GYA monitor operation. This feature is especially useful for the modification and extension of programs for all applications of the AN/GYA system.

The programming for the establishment of the link between the FMS system and the AN/GYA monitor has progressed to the stage that first experiments on operation are expected toward the end of July.

A dynamic allocation scheme of tapes has been included, which permitted multiprocessing of all AN/GYA programs with SORT/MERGE operation. This mode of operation was not possible, in the past, for some of the applications (e.g., the statistical language).

TASK IV INVESTIGATION OF COMPUTER CONSOLE INPUT AND DISPLAY

This task is concerned with a study of graphic devices for input and output in a man-computer interface. During the period under review, emphasis has been laid on a study of methods of processing graphic and text data for visual presentation. In context with the earlier works reviewed, the theories of languages for picture processing are being investigated. A special study is being carried on into the presentation of "display" mathematical workings.

4.1 Processing of Graphic and Text Data

4.1.1 Bunker-Ramo Corporation - Professor Glenn J. Culler

The early work of Culler and Fried in California indicates a certain success in the provision of on-line console facilities for scientists. Certain psychological insights obtained by them may prove of general value in the design of consoles, viz: It is found that continuous console operation for less than one hour or more than two hours tends to provide the user rather unsatisfactory results. Also a computer reaction time of less than 1/10 second is unnecessary; however, a delay of more than 1/10 second begins to make the human operator uneasy.

The Culler report describes an on-line Computing Center for Scientific Problems. It is designed and programmed so that on the Interface, FUNCTIONS rather than NUMBERS are the elements on which the commands operate.

Graphical output is via 2-17 inch CRT's with line-drawing capability. Alphanumeric output is via an eight inch CRT and a Flexowriter.

Thirty COMMAND buttons are provided which can be defined directly or recursively or by keyed-in programming, with the further possibility of multiplying the number of available commands through a system of "overlay" - each overlay being called just like a COMMAND.

The FUNCTIONS in the machine are represented by a maximum of 101 samples (100 intervals). Thus the equivalent of an "accumulator" can be thought of as 101 memory words. Twenty-four buttons are provided in order to identify function storage shelves and a final set of buttons provides the arithmetic +, -, 0, 1...9.

Because the operator can define new "commands" (e.g., SINE, COTH INVERSE, etc.) at the keyboard, he finds it "possible to build a representation, in the computer, of those analytical tools he believes valuable for a particular problem or problem area.... he is able, using only the concepts of classical mathematics, to create his own machine language, one tailor-made to his own needs."

A further report on this work will be given after some clarifications have been received.

4.1.2 The Algorithmic Theory of Language presented by Ross and the List-Structure approach used by Evans and others are found to overlap. Most practical approaches to picture processing have been found, however, to use rather specialized mechanisms and it is believed that it will be some time before a generally suitable formal theory is developed. This area of Study and Investigation does not yield rapidly to a clear understanding of the processes at work and since it is in the greatest state of flux, will need more and continuing work.

4.1.3 "Display" Mathematical Workings

Console text and graphical output systems have continuously increased their capability from alphanumeric (5 x 7 bit) characters to lines, circles, conics and more complex figures. However, the rather important area of presenting mathematical workings has not yet found any solution. Approaches that are being studied by us include the "Teager Table" system of analog-digital representation (a working model will be inspected shortly when it has been installed at MIT) and the work of Minsky and his students.

Mathematical display equations are highly stylized devices for communicating concepts and relationships. The optical, psychological and operational bases for "standard" representation are being investigated through a study of American Mathematical Society and A.I.P. practice, as well as of such standard publications as the Proceedings of the Royal Society, ZAMM, Journal of the SIAM, etc. Obviously much of what is accepted today as "standard practice" is due to the exigencies of typewriters, typesetting machines and printer's type. Not all these conventions are valid or useful for CRT displays.

As yet, very little work is published in this area and hence reliance is laid on personal discussions with other workers and our own detailed study of certain classes of problems in mathematical display.

4.2 Systems Considerations. The Engelbart Report

The man-machine information system at the Stanford Research Institute is based on the philosophy of a report which projects lines of research in the field of Text and Graphic on-line man-computer interaction. This is the report entitled "Augmenting Human Intellect - A Conceptual Framework" by D. C. Engelbart (Stanford Research Inst.)

In this work, Engelbart finds a perspective within which man-machine interaction may effectively be viewed. By analyzing intellectual activity itself and determining its basis in linguistic structure and symbol manipulative skills, he develops a hierachy of augmentation systems. Below we shall paraphrase his analysis of human intellectual activity and list some of the capabilities envisioned for a well-developed augmentation system.

Stages of Development

a. Concept Manipulation

The ability to "visualize" abstractions and conceptions. Concepts are non-verbalized and unprocessed.

b. Symbol Manipulation

Important step toward the ability to think in symbols rather than specific concepts. Emphasis here is on internal storage and manipulation, not communication. Example: A shepherd can keep track of his flock by counting sheep rather than by recognizing them. This amounts to having a crude, internal, language.

c. Manual, External Symbol Manipulation.

The next stage is the facility to store, externally, the symbols used.

Rate and Direction of Evolution in Thinking

There are theories that class languages among "self-organizing systems", where over a period of time quite subtle relationships among interacting elements can significantly influence the evolution of the system.

Korzybski and Whorf have argued that the language we use affects our thinking to a considerable extent. They say that lack of words for some types of concepts makes it hard to express these concepts. This leads to a Whorfian Hypothesis: The world-view of a culture is limited by the structure of the language which that society uses. Engelbart offers a Neo-Whorfian Hypothesis based on the preceding discussion.

Both the language use by a culture and the capability for intellectual activity are directly affected during their evolution by the means with which individuals control the external manipulation of symbols.

The Neo-Whorfian Hypothesis suggests that using a computer for manipulation leads to a fourth stage of development in Thinking.

d. Augmented External Symbol Manipulation distinguished by the Very Rapid Rate of manipulation with a Minimum Amount of Information supplied by the human.

In this stage, both the types of manipulation and the rate of creation of new symbols and new formats in which they can be manipulated may be expected to be considerably greater than before.

Specific Graphic Capabilities of a Computer

Engelbart visualizes an augmentation system of the future in operation, describing in general terms the capabilities of the computer and interface. These may be divided into two general classes -- systems for accepting symbol structures from a human operator, and systems for enabling the human to manipulate symbol structures in an augmented way.

As aids to feeding-in text, the following facilities are visualized:

2 Display Screens - Used horizontally, rather like drafting tables. One shows the input string and the other, a feedback, provides channel from the computer (see below).

2 Lightpens - One for each hand or screen; used in conjunction with keysets.

2 Keysets - One for each hand.

The keys are not merely alphanumeric, but allow for additional symbols and for use in a "shorthand" mode for rapid insertion of text. Single keys for bigrams and trigrams such as "ed" and "ing" provide speed without resorting to phonetic shorthand.

The keys can also be used to call for dictionary entries, synonyms, antonyms, etc. and to attach abbreviated labels to long strings.

Keys can also be used to structure input text in other forms than a simple string; for example, in developing an argument, the points would be listed vertically.

Artifact processes to aid in the manipulation of alphanumeric text are visualized as functioning much as a "copy editor".

Functions provided by lightpen-keyset combinations are designed to provide flexibility to the user in changing and d[illegible]ing ideas, argument-chains, etc.

Delete Word/Character; Insert Word/Character; Re-Justify are the more obvious "proofr[illegible] services. On a higher level, are Change Paragraph Break; Delete String; Move String to New Position (string to be moved and new location indicated by lightpen); Re-Adjust Margin; Change to Double Column Format; Adjust Column Widths so that two parallel columns are equal length; Change Structure of text presentation from string to nodal to lattice, etc.

It is of interest to note here that, allowing for cruder lightpen facilities, the lightpen-console Edit program developed by Inforonics performs many of the functions described above, and is structured to permit all of them.

Variable column width; Re-justification; Delete Character/Word/Sentence; Insert Character/String, etc. are already integral to the Inforonics Edit/Display program, for which a special keysetole was developed. A larger memory than provided by the PDP-1 would enable provision of the shorthand, labelling of strings and dictionary facilities.

Work performed on man-machine information systems at Stanford Research Institute following the Engelbart Report will be covered next, on the basis of the SRI Report of November 1963 and the Users' Guide for the SRI system as of April 1964. Facilities developed appear to be parallel to those developed by Inforonics with some additional sophistication. In particular, the indexing and storage of personal files for rapid access and display has received special attention.

4.3 Bibliography

1. American Institute of Physics. Author's Style Manual.

2. American Mathematical Society. Manual for Authors.

3. Billups, R. R., Rotstein, J. and Gardner, W. L. Solid State Display Panels with Photoconductive Control. Lincoln Laboratory Report No. 223.

4. Evans, T. G., Use of List-Structured Descriptions for programming manipulations on Line Drawings. (Unpublished Memorandum) A. F. C. R. L.

5. Evans, T. G., Heuristic Programs to Solve Geometric Analogy Problems (Unpublished Memorandum) A. F. C. R. L.

6. Kirsch, R. A., Symposium on Automatic Processing of Illustrated Text. National Bureau of Standards Report 8144.

7. William Byrd Press. Mathematics in Type. Richmond, Virginia, 1964.

8. Zieman, A. E. and Underwood, D. I., Remote Display Console for Computer Processed Data. Lincoln Laboratory Report No. 237.

9. Ross, D. R., An Algorithmic Theory of Language, Report ESL-TM-156, MIT, 1962.

TASK V INVESTIGATION OF AUTOMATED PROGRAM DEBUGGING TECHNIQUES

This task is concerned with a study of methods and development of computer programs which permit debugging of programs or program segments from a console.

Temporary reassignment of programmers during March and April caused an interruption of progress on this task. During the last month covered by this report, programming has been resumed. The coding for the facilities described on pages 40-43 of Quarterly Report No. 1 is now actively under way. These facilities are expected to be ready for testing in early July.

APPENDIX I

NONCENTRAL STATISTICAL DISTRIBUTION PROGRAMS FOR A COMPUTER LANGUAGE*

Rolf E. Bargmann, Sakti P. Ghosh

IBM Watson Research Center
Yorktown Heights, New York

ABSTRACT: Some of the numerical analysis problems have been discussed in connection with evaluating the incomplete probability integrals and also the quantities of commonly used statistical non-central distributions, e.g., noncentral chi-square, noncentral Beta, noncentral F and noncentral t distributions. The methods most suitable for digital computers from the point of view of computer time and accuracy have been discussed. FORTRAN routines for evaluating the incomplete probability integrals and quantities of these noncentral distributions have been developed and are given in this report

* This work was supported in part by Air Force Research Contract AF19(602)-3303

1. INTRODUCTION

In an earlier report (3), the authors discussed some numerical techniques used in the development of statistical distribution programs for a statistical computer language. Included in this report were Fortran programs for the Normal, Gamma, Beta, Student's t, and F distributions. The present report extends the discussion to the noncentral forms for these distributions and includes the corresponding Fortran programs.

Though the noncentral distributions are very useful in statistical analysis yet very little attempt has been made to develop good programs because of the complex nature of the noncentral distributions. The difficulties of programming are greatly increased when high degree of accuracy and wide range of admissibility of the parameters are required. To achieve a desired degree of accuracy certain boundary conditions had to be imposed. These are:

(1) All the parameters of the noncentral distributions were restricted to the range 10^{-8} to 10^{8}. (Accuracy of the results outside this range becomes doubtful due to the countless arithmetic operations involved in the calculations.)

(2) Regardless of the choice of the input (parameter values, abscissa, or probability) the results were required to be

correct to at least five significant digits. Due to this requirement the calculation time for some of the routines may be lengthy (2 or 3 minutes) when the value of the noncentrality parameter is large.

Various approaches for evaluating noncentral distributions by approximate formulae have been suggested in the literature (8), (10) but upon investigation it was found that most of these give no more than two-place accuracy for values of noncentrality parameters as large as 50. Since programs for central distributions had already been developed by the authors (3), an attempt was made to make use of the fact that a noncentral distribution can often be expressed as an infinite sum of weighted central distributions, with the weights being Poisson terms. It was discovered, however, that with the latter approach the programs consumed too much time and it was extremely difficult to maintain sufficient accuracy. Fortunately, a simple modified technique for evaluating the infinite sum greatly increased the accuracy and decreased the computation time. This approach is used in the present report to evaluate the noncentral χ^2 and noncentral Beta distributions. It is also shown that these results can also be used to evaluate the noncentral Gamma, F, and t distributions.

2. NONCENTRAL CHI-SQUARE DISTRIBUTION

The density function of the noncentral χ^2 variate is given by (2.1)

$$f(\lambda, m, y) = \sum_{j=0}^{\infty} \frac{e^{-\lambda/2}(\lambda/2)^j}{j!} \quad \frac{e^{-y/2}\, y^{m/2+j-1}}{2^{m/2+j}\ \Gamma(m/2+j)} \qquad (2.1)$$

where λ is the noncentrality parameter and m is the degrees of freedom. The routine developed for evaluating the incomplete probability integral of the noncentral χ^2 is defined by

NCHIX (X, DF, C, P, Z) (2.2)

where X, DF, C, P, and Z are, respectively, matrices (all of the same size) of abscissas, degrees of freedom, noncentrality parameters, probabilities, and ordinates. The first three are input matrices, while P and Z are output matrices. For each x in X the routine (2.2) calculates the probability

$$p = \int_0^x f(\lambda, m, y)\, dy \qquad (2.3)$$

and the ordinate

$$z = f(\lambda, m, y)$$

with m, λ, p and z being the elements of DF, C, P, and Z, respectively, which are in the same position as x in X.

The inverse routine is given in (2.4)

$$\text{NCHIP (P, DF, C, X)} \tag{2.4}$$

This routine evaluates the abscissa (quantities) for a set of input matrices, P (probabilities), DF (degrees of freedom), and C (noncentrality parameters). The result is a matrix of quantities, X, each element of which is associated with the corresponding elements of the input matrices.

The routines (2.2) and (2.4) require the direct or inverse evaluation of:

$$J_x(\lambda, m) = \int_0^x f(\lambda, m, y)\, dy$$

$$= \sum_{j=0}^{\infty} P(\lambda/2, j)\, J_x(m+2j) \tag{2.5}$$

where

$$P(\lambda/2, j) = e^{-\lambda/2} (\lambda/2)^j \;/\; \Gamma(j+1) \tag{2.6}$$

and

$$J_x(m+2j) = \int_0^x \frac{e^{-y/2}}{2^{(m+2j)/2}} \frac{y^{(m+2j-2)/2}}{\Gamma((m+2j)/2)}\, dy \tag{2.7}$$

A method for evaluating $J_x(m)$ has been discussed in detail by the authors (3), however, a faster method based on continued fraction will be discussed here. This method is due to Laplace and is given in

(4). A slight modification of Laplace's form can be stated as

$$\frac{1}{\Gamma(a)} \int_z^{\infty} e^{-u} u^{a-1} \, du = \frac{e^{-z} z^a}{\Gamma(a)} \quad \frac{1}{z+} \; \frac{1-a}{1+} \; \frac{1}{z+} \; \frac{2-a}{1+}$$

$$\frac{2}{z+} \; \frac{3-a}{1+} \; \frac{3}{z+} \; \cdots\cdots\cdots \tag{2.8}$$

$J_x(m+2j)$ may be evaluated by substituting $a = m/2 + j$, $z = x/2$ in (2.8) and subtracting the result from unity. By using fifty terms in the continued fraction it is possible to obtain at least 12 decimal places accuracy. Unfortunately, the continued fraction approach cannot be used for the entire range of the parameters. It was found that for the range of the parameters given by (2.9)

$$4 < m + 2j < 1000 \text{ and } x > m-2 \text{ and } z > .005 \tag{2.9}$$

it was appropriate to evaluate $J_x(m + 2j)$ by using continued fraction. Also, when $m + 2j > 1000$ the Hilferty-Wilson (12) transformation proved to be more effective, while in the remaining range of the parameters the original procedure presented by the authors (3) are more efficient. Additionally, when the noncentrality parameter is larger than 2000 ($\lambda \geq 2000$) the Hilferty-Wilson transformation has been used to calculate $P(\lambda/2, j)$.

In evaluating the infinite sum (2.5) on the computer it was observed that it was more convenient to start the summation at the modal value $j = [\lambda/2]$ and accumulate terms on either side by increasing and decreasing j until the terms become smaller than the desired degree of accuracy or j becomes zero. This method of building the infinite sum by starting at the modal term has been used in all the other noncentral distributions discussed in this paper.

In evaluating the inverse routine, Newton's method (9) of solving equations has been used and if Newton's method failed to converge then Horner's method has been used. For the choice of the initial point it has been found that $m+\lambda-2$ is quite satisfactory.

3. NONCENTRAL BETA DISTRIBUTION

The density of the noncentral beta distribution is given by

$$f(\lambda, m, n, y) = \sum_{j=0}^{\infty} \frac{e^{-\lambda/2} (\lambda/2)^j}{j!} \frac{y^{m+j-1} (1-y)^{n-1}}{B(m+j, n)} \tag{3.1}$$

The routine for evaluating the incomplete probability integral and the ordinates of (3.1) is given by

NCBTX (X, DF1, DF2, C, P, Z) (3.2)

where all the arguments are matrices having the same meaning as in Section 2. Of course, now, two degrees of freedom matrices, DF1 and DF2, are needed. The first of these, DF1, involves the degrees of freedom, m, and the second, DF2, applies to n. The routine (3.2) computes for each value x of X and corresponding parameter values m, n, and λ (i.e., elements of DF1, DF2, and C in the same position as x is in X), the probability, p, and ordinate z (the corresponding elements of P and Z). The value of z is the ordinate of (3.1) and p is defined by

$$p = I_x(\lambda, m, n) = \int_0^x f(\lambda, m, n, y)\, dy \tag{3.3}$$

The inverse routine given by

$$\text{NCBTP (P, DF1, DF2, C, X)} \tag{3.4}$$

computes the quantities, x, for a given probability, p, and specified parameter values m, n, and λ, where these symbols represent the same quantities as defined for (3.2).

For evaluating the routine (3.2) the incomplete probability integral was written in the form

$$I_x(\lambda, m, n) = \sum_{j=0}^{\infty} P(\lambda/2, j)\; I_x(m+j, n) \tag{3.5}$$

where $P(\lambda/2, j)$ is defined in [illegible] and

$$I_x(m+j, n) = \frac{\int_0^x y^{m+j-1} (1-y)^{n-1} \, dy}{B(m+j, n)} \tag{3.6}$$

is the incomplete probability integral of a central beta distribution. (3.6) may be evaluated by the technique discussed by the authors (3). When the parameters m and n are both in the neighborhood of unity (specifically, when each is within 1 ± 10^{-8}) (3.5) reduces to very simple forms, which can be used directly instead of going to the general procedure. When $\lambda = 0$ (specifically, $0 \leq \lambda \leq 10^{-8}$), $m \approx 1$, $n \approx 1$, then $I_x(\lambda, m, n) \approx x$ and $f(\lambda, m, n, x) \approx 1$. Again, when $\lambda \neq 0$ (specifically $\lambda > 10^{-8}$), $m \approx 1$, $n \approx 1$, then $I_x(\lambda, m, n) \approx x \operatorname{Exp}\left[(x\lambda - \lambda)/2\right]$ and $f(\lambda, m, n, x) = (1 + x\lambda/2) \operatorname{Exp}\left[(x\lambda-\lambda)/2\right]$. For developing the infinite sum the same method as discussed in section 2 has been used.

For the inverse routine (3.4) the same method of numerical solution of an equation as discussed in Section 2 was used. The problem of choice of a good starting point (x_0) was solved by trial and error. The following values of x_0 given in (3.7) seem to work satisfactory

$$\begin{aligned} &\text{For } p < .95, \quad x_0 = (m-1+\lambda/2) \;/\; (m+n-2+\lambda/2) \\ &\text{For } p \geq .95, \quad x_0 = (m-1+3\lambda/2) \;/\; (m+n-2+3\lambda/2) \end{aligned} \tag{3.7}$$

4 NONCENTRAL F DISTRIBUTION

The density function for the noncentral F distribution is given by

$$f(\lambda, m, n, y) = \sum_{j=0}^{\infty} \left[\frac{e^{-\lambda/2}(\lambda/2)^j}{j!} \left(\frac{m}{n}\right)^{m/2+j} \frac{y^{m/2+j-1}}{(1+my/n)^{(m+n+2j)/2}} \times \frac{1}{B(m/2+j, n/2)} \right] \tag{4.1}$$

The routine for evaluating the incomplete probability integral and the ordinates of (4.1) is

$$\text{NCFX (X, DF1, DF2, C, P, Z)} \tag{4.2}$$

where the arguments are matrices and have been defined in Section 3. The routine (4.2) computes for each value x of X and corresponding parameter values m, n, λ (i.e., elements of DF1, DF2, and C in the same position as x in X), the probability, p, and the ordinate, z.

The inverse routine, given by

$$\text{NCFP (P, DF1, DF2, C, X)} \tag{4.3}$$

computes the quantities x, for given probabilities and parameter values (similar to the inverse routine defined in the previous section, but for F distribution).

It is well known that the transformation

$$(my/n) \;/\; (1+ my/n) \tag{4.4}$$

transforms the noncentral F-distribution to noncentral Beta distribution with parameters m/2 and n/2. Thus,

$$I_F(m, n) = \int_0^F f(\lambda, m, n, y)\, dy$$

$$= \int_0^{F_o} \frac{x^{m/2+j-1}(1-x)^{n/2-1}}{B(m/2+j, n/2)}\, dx \tag{4.5}$$

where $F_o = (mF/n)/(1 + mF/n)$. Thus, (4.2) was obtained by making the necessary changes in (3.2). For evaluating the ordinate of (4.1) the ordinate of noncentral beta has to be multiplied by m/n.

The inverse routine (4.3) was obtained from (3.4) by first calculating the Beta value for given values of p, m, n, and λ and then making the necessary transformation to get the F-value.

5. NONCENTRAL T DISTRIBUTION

The density of the noncentral t-distribution is

$$f(\lambda, N, y) = \sum_{j=0}^{\infty} P(\lambda/2, j/2) \frac{(y^2/N)^{j/2}}{\sqrt{N}\, B(\frac{j+1}{2}, \frac{N}{2})(1+\frac{y^2}{N})^{\frac{N+j+1}{2}}} \tag{5.1}$$

The routine

$$\text{NCTX (T, DF, C, P, Z)} \tag{5.2}$$

will calculate the incomplete probability integral of the noncentral t-distribution. The arguments are matrices of input and output parameters. For a given t of T the routine will calculate the incomplete probability integral

$$p = I_t(\lambda, N, t) = \int_{-\infty}^{t} f(\lambda, N, y)\, dy \tag{5.3}$$

where N, λ, and p are the corresponding elements of DF, C, and P matrices. The routine will also calculate the ordinate z given by (5.1).

The inverse routine

$$\text{NCTP (P, DF, C, T)} \tag{5.4}$$

calculates the quantities, t, for given p, N, and λ which are the elements of P, DF, and C, respectively.

There are a few different methods for evaluating the incomplete probability integral of the noncentral t-distribution, but as the aim of the present system of programs was to make use of the routines already developed, hence, the method based on incomplete central beta distribution was used. It is easy to see (4) that (5.3) can be written as

$$F_t(\lambda, N, t) = \frac{1}{2}\Bigg[1 - \sum_{j=0}^{\infty} P(\lambda/2, j + \tfrac{1}{2})$$

$$+ \sum_{j=0}^{\infty} P(\lambda/2, j+\tfrac{1}{2})\; I_{T_o}(j+1, N/2)$$

$$\pm \sum_{j=0}^{\infty} P(\lambda/2, j)\; I_{T_o}(j + \tfrac{1}{2}, N/2)\Bigg] \tag{5.5}$$

where $T_o = (t^2/N) / (1+t^2/N)$ (5.6)

$I_{T_o}(j + 1, N/2)$ and $I_{T_o}(j + \frac{1}{2}, N/2)$ are the incomplete Beta functions defined in (3.6). The sign in front of the last term in (5.5) is the same as the sign of t. The ordinates (5.1) can also be calculated from the ordinates of a central Beta distribution. On simplification, (5.1) may be reduced to

$$f(\lambda, N, t) = \sum_{j=0}^{\infty} P(\lambda/2, j)\; \frac{T_o^{1/2}\,(1-T_o)^{3/2}}{\sqrt{N}}\;\; \theta(T_o, j+\tfrac{1}{2}, N/2)$$

$$\pm \sum_{j=0}^{\infty} P(\lambda/2, j+\tfrac{1}{2})\; \frac{T_o^{1/2}\,(1-T_o)^{3/2}}{\sqrt{N}}\;\; \theta(T_o, j+1, N/2) \tag{5.7}$$

where $\theta(T_o, j+\frac{1}{2}, N/2)$ and $\theta(T_o, j+1, N/2)$ are the ordinates of a central beta distribution at T_o for parameters $(j+\frac{1}{2}, N/2)$ and $(j+1, N/2)$, respectively.

Now (5.2) can be evaluated very easily from (5.5) and (5.7) by using the same technique as used for evaluating (3.2).

The inverse routine (5.4) has been evaluated by using Newton's method of solving equations, on the T_o -scale and then making the proper transformation to obtain t. The starting point (x_o) for Newton's method was determined by trial and error and the following worked quite satisfactory.

$$
\begin{aligned}
x_o &= 1-p, \text{ for } |p| \leq 10^{-5} \\
x_o &= 1-p, \text{ for } 1-p \leq 10^{-5} \\
x_o &= |p-p_1|, \text{ for } |p-p_1| \leq 10^{-5} \\
x_o &= p, \quad \text{otherwise}
\end{aligned}
\tag{5.8}
$$

where $p_1 = \left[1 - \sum_{j=0}^{\infty} P(\lambda/2, j+\frac{1}{2})\right] / 2$

6. ACKNOWLEDGEMENT

The authors are grateful to Dr. Harry O. Posten of the University of Connecticut for his valuable suggestions on the style of writing the paper in its present form.

7 REFERENCES

1. ABDEL-ATY, S. H. (1954). Approximate formulae for percentage points and the probability integral of the noncentral χ^2 distribution. Biometrika, 41, p. 538-540

2. AROIAN, L. (1941). Continued fraction for the incomplete beta function. Ann. Math. Stat., 12, p. 218-223.

3. BARGMANN, R. E. and GHOSH, S. P. (1963). Statistical distribution programs for a computor language. IBM Research Report No. 1094.

4. BOWKER, A. H. and GOODE H. P. (1952). Sampling inspection by variables. (p. 127). McGraw-Hill Book Company, Inc., New York.

5. CRAIG, C. C. (1941). Note on the distribution of noncentral t with an application. Ann. Math. Stat., 12, p. 224-228.

6. FIX, E., HODGES, J. L. (Jr.); LEHMANN, E. L. (1961). The restricted chi-square test. Harold Cramér Volume (p. 92-107). Wiley Publications, New York.

7. LAUBSCHER, N. (1960). Normalising the noncentral t and F distributions. Ann. Math. Stat., 31, p. 1105-1112.

8. PATNAIK, P. B. (1949). The noncentral χ^2 and F-distributions and their applications. Biometrika, 36, p. 202-232.

9. SCARBOROUGH, J. B. (1958). Numerical mathematical analysis. Johns Hopkins Press, Baltimore.

10. SEBER, G. A. F. (1963). The noncentral chi-square and beta distributions. Biometrika, 50, p. 542-545.

11. WALL, H. S. (1948). Analytic theory of continued fraction. (p. 356). Van Nostrand Co., New York.

12. WILSON E. B. and HILFERTY, M. M. (1931). The distribution of chi-square. Proc. Nat. Acad. Sci., 17, p. 684-688.

8. FORTRAN PROGRAMS

In this section the FORTRAN programs for the routines discussed in the previous sections are presented. These programs are written in double precision as subroutines. They can be run without modifications on an IBM 7090 computer.

```
      SUBROUTINE NCHIX(X2,AG,BG,PG,ORG)
      DIMENSION X2(1),AG(1),BG(1),PG(1),ORG(1)
      COMMON NRA,NCA,NRB,NCB,NRC,NCC,NRD,NCD
      DO 201 K15=1,NCA
      DO 201 K12=1,NRA
      KA=(K15-1)*NRA+K12
D     FLAM=0.
D     FIRES=0.
D     FIORD=0.
D     G2=0.
D     P=0.
D     ORD=0.
D     X=0.
      G2=AG(KA)
      FLAM=PG(KA)
D     FLAM=FLAM/2.
      X=X2(KA)
      IF(G2) 199,199,100
  100 IF(X) 199,90,90
   90 IF(X-1.E-8) 171,171,101
D 171 P=0.
      IF(G2-2.) 164,165,166
D 164 ORD=.99999999E30
      GO TO 200
  165 ORD=1.
      GO TO 200
  166 ORD=0.
      GO TO 200
  101 LLX=FLAM
D     FLLX=LLX
D     AB1=G2+2.*FLLX
D     G2=AB1+2.
      LX=FLAM
      LX=LX+1
  441 LX=LX-1
D     AA1=0.
      AA1=LX
D     G2=G2-2.
      INDEX=1
      IF(LX) 455,454,454
  455 LLX=LLX+1
      LX=LLX
D     AA1=0.
      AA1=LLX
D     G2=AB1+2.
D     AB1=G2
      INDEX=2
D 454 P=0.
D     PP=0.
D     RESULT=0.
      IF(G2-1000.) 168,170,170
  168 IF(X-2000.) 167,169,169
D 169 P=1.
D     ORD=0.
      GO TO 200
```

```
  170 Y1=LOGF(X/G2)/3.
      Y1=EXPF(Y1)
      Y2=1.-2./(9.*G2)
      Y3=SQRTF(2./(9.*G2))
      XX=(Y1-Y2)/Y3
      CALL NORM9(XX,PRO,ORD)
D     P=PRO
D     ORD=ORD
      GO TO 200
  167 IF(G2-4.) 135,135,136
  135 G11=G2/2.+5.E-8
      K=XINTF(G11)
D     THETA=0.
      THETA=G2/2.-FLOATF(K)
      IF(THETA-1.E-7) 145,145,146
  145 THETA=0.
  146 CONTINUE
D     A=THETA*LOGF(X)-X/2.-(1.+THETA)*LOGF(2.)-ZLOGGM(1.+THETA)
D     A3=A
      IF(A+80.) 103,103,102
D102  A2=EXPF(A)
D     T3=A2
D     ORD2=A2
D     T2=0.
      IF(THETA) 130,130,131
D130  A3=A3-LOGF(X)
      GO TO 132
D131  A3=A3+LOGF(2.)+LOGF(THETA)-LOGF(X)
  132 IF(A3+80.) 109,109,108
  108 IF(A3-80.) 162,162,163
  163 ORD1=.99999999E30
      GO TO 104
D162  A2=EXPF(A3)
D     ORD1=A2
      GO TO 104
D109  ORD1=0.
      GO TO 104
D103  T3=0.
D     ORD2=0.
D     ORD1=0.
D     T2=0.
  104 I=1
  105 I=I+1
D     XI=I
D     A=(XI+THETA)*LOGF(X/2.)-LOGF(X)-X/2.-ZLOGGM(XI+THETA)
      IF(A+80.) 107,107,106
D106  A2=EXPF(A)
D     ORD3=A2
D     T2=T2+A2
  107 IF(I-K) 110,111,111
  110 GO TO 105
  111 IF(THETA) 138,138,139
  139 IF(X-5.) 148,148,149
  148 I=1
D     T11=1.
```

```
      N=0
D     A=LOGF(X/2.)+LOGF((1.+THETA)/(2.+THETA))
      IF(A+80.) 113,113,112
D112  T11=T11-EXPF(A)
 113  J=-1
 114  I=I+1
D     XI=I
      J=-1*J
D     A3=(1.+THETA)/(XI+THETA+1.)
      IF(A3-1.E-8) 143,143,147
D147  A=XI*LOGF(X/2.)-ZLOGGM(XI+1.)+LOGF(A3)
      IF(A+80.) 143,143,144
D143  T12=0.
      GO TO 119
D144  T13=EXPF(A)
      C=FLOATF(J)
      T12=SIGNF(T13,C)
D     T11=T11+T12
      N=N+1
  115 IF(N-50) 116,117,117
 116  GO TO 114
 117  PRINT 118,K15,K12
 118  FORMAT(13H0 IN ELEMENT I4,I4,61H FULL CONVERGENCE WAS NOT ATTAINED
     1. MAY BE SOMEWHAT UNPRECISE)
 119  IF(T11) 133,133,134
 133  GO TO 121
D134  A=(1.+THETA)*LOGF(X/2.)+LOGF(T11)-ZLOGGM(2.+THETA)
      IF(A+80.) 121,121,120
D120  T1=EXPF(A)
      GO TO 122
D121  T1=0.
      GO TO 122
D138  A=-X/2.
      IF(A+80.) 140,140,141
D140  A2=0.
      GO TO 142
D141  A2=EXPF(A)
D142  T1=1.-A2
      GO TO 122
 149  N=25
      A2=0.
      I=0
 153  I=I+1
D     XI=I
D     A=-(13.*X)/XI+(THETA+1.)*LOGF(13.*X/XI)-ZLOGGM(1.+THETA)-LOGF(XI)
      IF(A+80.) 150,150,151
D150  T11=0.
      GO TO 152
D151  T11=EXPF(A)
D152  A2=A2+T11
      IF(I-N) 153,154,154
D154  B=1.0128205+THETA/156.-X/312.
      IF(B) 158,155,[illegible]
D158  B=ABSF(B)
D     A=-X/2.+(THETA+1.)*LOGF(X/2.)+LOGF(B)-LOGF([illegible]) [illegible]LOGGM(THETA+1.)
```

```
      IF(A+80.) 155,155,161
D161  C=-EXPF(A)
      GO TO 157
D160  A=-X/2.+(THETA+1.)*LOGF(X/2.)+LOGF(B)-LOGF(52.)-ZLOGGM(THETA+1.)
      IF(A+80.) 155,155,156
D155  C=0.
      GO TO 157
D156  C=EXPF(A)
D157  D=A2+C
D     T1=1.-D
 122  IF(G2-2.) 123,124,124
D123  RESULT=2.*T3+T1
D     ORD=ORD1
      GO TO 196
 124  IF(G2-4.) 125,126,126
D125  RESULT=T1
D     ORD=ORD2
      GO TO 196
D126  RESULT=T1-2.*T2
D     ORD=ORD3
 196  IF(RESULT) 194,194,195
D 194 P=0.
      GO TO 200
D 195 P=RESULT
      GO TO 200
D 136 X=X/2.
D     G2=G2/2.
D     ORDL=-X+(G2-1.)*LOGF(X)-ZLOGGM(G2)
      IF(ORDL+60.) 172,172,173
D 172 ORD=0.
      IF(X-G2+1.) 174,175,175
D 174 P=0.
      GO TO 200
D 175 P=1.
      GO TO 200
D 173 ORDA=EXPF(ORDL)
D     ORD1=ORDA/2.
D     ORD=ORD1
      IF(ORD1-.005) 178,178,179
  178 IF(X-G2+1.) 180,180,179
D 180 X=X*2.
D     G2=G2*2.
      GO TO 135
  179 II=50
D     XI=II
D     SS=X+(XI-G2)/XI
  176 II=II-1
D     XI=II
D     SS=X+(XI-G2)/(1.+XI/SS)
      IF(II-1) 177,177,176
D 177 SS=X/SS
D     PROB=ORDA*SS
D     PROB=1.-PROB
D     P=PROB
D     X=X*2.
```

```
D     G2=G2*2.
 200  CONTINUE
      IF(FLAM) 199,442,442
 442  IF(FLAM-1.E-7) 443,443,444
  443 PG(KA)=P
      ORG(KA)=ORD
      GO TO 201
 444  CFLAM=1000.
      IF(FLAM-CFLAM) 447,448,448
 448  PRINT 449,K12,K15
 449  FORMAT(65H0 NORMAL APPROXIMATION WAS USED FOR POISSON PART OF NONC
     1ENTRAL F 2I9)
      BA=(AA1/FLAM)**.33333333-1.+2./(9.*FLAM)
      BB=BA/SQRTF(2./(9.*FLAM))
      CALL NORM9(BB,CC,PP1)
      PP=PP1
      GO TO 401
D447  C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
      IF(C1+45.) 450,450,451
D450  PP=0.
      GO TO 401
D451  PP=EXPF(C1)
 401  CONTINUE
D     SUBRES=P*PP
D     SUBORD=PP*ORD
D     FIORD=FIORD+SUBORD
D     FIRES=FIRES+SUBRES
      IF(SUBRES-1.E-12) 456,456,457
  457 GO TO(441,455),INDEX
  456 GO TO(455,452),INDEX
  452 PG(KA)=FIRES
      IF(FIORD-.99999999E30) 18,19,19
   19 ORG(KA)=.99999999E30
      GO TO 201
   18 ORG(KA)=FIORD
      GO TO 201
  199 PRINT 198,K15,K12
  198 FORMAT(20H0 ARGUMENT NEGATIVE 14,14)
      PG(KA)=-0.
      ORG(KA)=-0.
  201 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NCHIP(PG,AG,BG,AX)
      DIMENSION PG(1),AG(1),BG(1),AX(1)
      COMMON NRA,NCA,NRB,NCB,NRC,NCC,NRD,NCD
      DO 300 K15=1,NCA
      DO 300 K12=1,NRA
      KA=(K15-1)*NRA+K12
D     FLAM=0.
D     AA=0.
D     PP=0.
D     PA=0.
D     G2=0.
      PP=PG(KA)
      G2=AG(KA)
      FLAM=BG(KA)
D     SG2=G2
D     FLAM=FLAM/2.
      IF(PP) 299,225,225
  225 IF(PP-1.E-8) 226,226,202
  226 AX(KA)=0.
      GO TO 300
  202 IF(PP-1.) 224,221,299
  221 AX(KA)=.99999999E30
      GO TO 300
  224 IF(G2) 299,299,203
D 203 X=G2+2.*FLAM-2.+.01
      IF(X) 207,207,208
D 207 X=0.1
  208 I=0
  209 I=I+1
D527  RESULT=0.
D     FIRES=0.
D     IORD=0.
D     G2=SG2
D     X=X
      LLX=FLAM
D     FLLX=LLX
D     AB1=G2+2.*FLLX
D     G2=AB1+2.
      LX=FLAM
      LX=LX+1
  441 LX=LX-1
D     AA1=0.
      AA1=X
D     G2=G2-2.
      ...=1
      IF(LX) 455,454,454
  455 X=LX+1
      ...=LLX
      ...A1=0.
      ...1=LLX
D     ...=AA1+2.
D     ...=G2
      ...=2
```

```
      IF(G2) 199,199,100
  100 IF(X) 199,90,90
   90 IF(X-1.E-8) 171,171,101
D171  PA=0.
      IF(G2-2.) 164,165,166
 164  ORD=.99999999999E30
      GO TO 200
D165  ORD=1.
      GO TO 200
 166  ORD=0.
      GO TO 200
  101 IF(G2-1000.) 168,170,170
 168  IF(X-2000.) 167,169,169
D169  PA=1.
D     ORD=0.
      GO TO 200
  170 Y1=LOGF(X/G2)/3.
      Y1=EXPF(Y1)
      Y2=1.-2./(9.*G2)
      Y3=SQRTF(2./(9.*G2))
      XX=(Y1-Y2)/Y3
      CALL NORM9(XX,PA,ORD)
      GO TO 200
  167 IF(G2-4.) 135,135,136
  135 G11=G2/2.+5.E-8
      K=XINTF(G11)
D     THETA=0.
      THETA=G2/2.-FLOATF(K)
      IF(THETA-1.E-7) 145,145,146
 145  THETA=0.
 146  CONTINUE
D     A=THETA*LOGF(X)-X/2.-(1.+THETA)*LOGF(2.)-ZLOGGM(1.+THETA)
D     A3=A
      IF(A+80.) 103,103,102
D102  A2=EXPF(A)
D     T3=A2
D     ORD2=A2
D     T2=0.
      IF(THETA) 130,130,131
D130  A3=A3-LOGF(X)
      GO TO 132
D131  A3=A3+LOGF(2.)+LOGF(THETA)-LOGF(X)
 132  IF(A3+80.) 109,109,108
 108  IF(A3-80.) 162,162,163
 163  ORD1=.99999999E30
      GO TO 104
D162  A2=EXPF(A3)
D     ORD1=A2
      GO TO 104
D109  ORD1=0.
      GO TO 104
D103  T3=0.
D     ORD2=0.
D     ORD1=0.
D     T2=0.
```

```
 104  I=1
 105  I=I+1
D     XI=I
D     A=(XI+THETA)*LOGF(X/2.)-LOGF(X)-X/2.-ZLOGGM(XI+THETA)
      IF(A+80.) 107,107,106
D106  A2=EXPF(A)
D     [illegible]=A2
D     T2=T2+A2
 107  IF(I-K) 110,111,111
 110  GO TO 105
 111  IF(THETA) 138,138,139
 139  IF(X-5.) 148,148,149
 148  I=1
D     T11=1.
      N=0
D     A=LOGF(X/2.)+LOGF((1.+THETA)/(2.+THETA))
      IF(A+80.) 113,113,112
D112  T11=T11-EXPF(A)
 113  J=-1
 114  I=I+1
D     XI=I
      J=-1*J
D     A3=(1.+THETA)/(XI+THETA+1.)
      IF(A3-1.E-8) 143,143,147
D147  A=XI*LOGF(X/2.)-ZLOGGM(XI+1.)+LOGF(A3)
      IF(A+80.) 143,143,144
D143  T12=0.
      GO TO 119
D144  T13=EXPF(A)
      C=FLOATF(J)
      T12=SIGNF(T13,C)
D     T11=T11+T12
      N=N+1
  115 IF(N-50) 116,117,117
 116  GO TO 114
 117  PRINT 118,K15,K12
 118  FORMAT(13HO IN ELEMENT I4,I4,61H FULL CONVERGENCE WAS NOT ATTAINE
     1. MAY BE SOMEWHAT UNPRECISE)
 119  IF(T11) 133,133,134
 133  GO TO 121
D134  A=(1.+THETA)*LOGF(X/2.)+LOGF(T11)-ZLOGGM(2.+THETA)
      IF(A+80.) 121,121,120
D120  T1=EXPF(A)
      GO TO 122
D121  T1=0.
      GO TO 122
D138  A=-X/2.
      IF(A+80.) 140,140,141
D140  A2=0.
      GO TO 142
D141  A2=EXPF(A)
D142  T1=1.-A2
      GO TO 122
 149  N=25
D     A2=0.
```

```
          I=0
  153     I=I+1
 D        XI=I
 D        A=-(13.*X)/XI+(THETA+1.)*LOGF(13.*X/XI)-ZLOGGM(1.+THETA)-LOGF(XI)
          IF(A+80.) 150,150,151
 D150     T11=0.
          GO TO 152
 D151     T11=EXPF(A)
 D152     A2=A2+T11
          IF(I-N) 153,154,154
 D154     B=1.01282051+THETA/156.-X/312.
          IF(B) 158,155,160
 D158     B=ABSF(B)
 D        A=-X/2.+(THETA+1.)*LOGF(X/2.)+LOGF(B)-LOGF(52.)-ZLOGGM(THETA+1.)
          IF(A+80.) 155,155,161
 D161     C=-EXPF(A)
          GO TO 157
 D160     A=-X/2.+(THETA+1.)*LOGF(X/2.)+LOGF(B)-LOGF(52.)-ZLOGGM(THETA+1.)
          IF(A+80.) 155,155,156
 D155     C=0.
          GO TO 157
 D156     C=EXPF(A)
 D157     D=A2+C
 D        T1=1.-D
  122     IF(G2-2.) 123,124,124
 D123     RESULT=2.*T3+T1
 D        ORD=ORD1
          GO TO 196
  124     IF(G2-4.) 125,126,126
 D125     RESULT=T1
 D        ORD=ORD2
          GO TO 196
 D126     RESULT=T1-2.*T2
 D        ORD=ORD3
  196     IF(RESULT) 194,194,195
 D194     PA=0.
          GO TO 200
 D195     PA=RESULT
          GO TO 200
 D 136    X=X/2.
 D        G2=G2/2.
 D        ORDL=-X+(G2-1.)*LOGF(X)-ZLOGGM(G2)
          IF(ORDL+60.) 172,172,173
 D 172    ORD=0.
          IF(X-G2+1.) 174,175,175
 D 174    PA=0.
          GO TO 200
 D 175    PA=1.
          GO TO 200
 D 173    ORDA=EXPF(ORDL)
 D        ORD1=ORDA/2.
 D        ORD=ORD1
          IF(ORD1-.005) 178,178,179
  178     IF(X-G2+1.) 180,180,179
 D 180    X=X*2.
```

```
D       G2=G2*2.
        GO TO 135
  179 II=50
D       XI=II
D       SS=X+(XI-G2)/XI
  176 II=II-1
D       XI=II
D       SS=X+(XI-G2)/(1.+XI/SS)
        IF(II-1) 177,177,176
D 177 SS=X/SS
D       PROB=ORDA*SS
D       PROB=1.-PROB
D       PA=PROB
D       X=X*2.
D       G2=G2*2.
 200  CONTINUE
        IF(FLAM) 199,442,442
  442 IF(FLAM-1.E-7) 201,201,444
 444  CFLAM=1000.
        IF(FLAM-CFLAM) 447,448,448
 448  PRINT 449,K12,K15
 449  FORMAT(65H0 NORMAL APPROXIMATION WAS USED FOR POISSON PART OF NONC
     1ENTRAL F 219)
        BA=(AA1/FLAM)**.33333333-1.+2./(9.*FLAM)
        BB=BA/SQRTF(2./(9.*FLAM))
        CALL NORM9(BB,CC,PP1)
        APP=PP1
        GO TO 401
D447  C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
        IF(C1+45.) 450,450,451
D 450 APP=0.
        GO TO 401
D 451 APP=EXPF(C1)
 401  CONTINUE
D       SUBRES=PA*APP
D       SUBORD=APP*ORD
D       FIORD=FIORD+SUBORD
D       FIRES=FIRES+SUBRES
        IF(SUBRES-1.E-12) 456,456,457
  457  GO TO(441,455),INDE
  456 GO TO(455,452),INDE
D 452 PA=FIRES
        IF(FIORD-.99999999E30) 18,19,19
D  19 ORD=.99999999E30
        GO TO 201
D  18 ORD=FIORD
  201 IF(II-1) 547,547,548
 547  INDEX=1
 548  GO TO(532,533,534),INDEX
 532  IF(ORD) 522,522,529
 529  CONTINUE
D       XC=X-(PA-PP)/ORD
D       DEL=0.
D       ABDP=ABSF(PA-PP)
D       ABDP=ABDP/PP
```

```
      IF(ABDP-1.E-8) 516,518,520
 518  AX(KA)=XC
      GO TO 300
 520  CONTINUE
 556  IF(XC-1.E-30) 558,559,559
D558  X=X/2.
      GO TO 560
D559  X=XC
 560  INDEX=1
      NN=15
      IF(II-NN) 209,209,522
 522  NCT=0
D     HX=X
      IF(X-1.E-30) 550,550,551
D550  X=.01
 551  CONTINUE
 552  IF(PA-PP) 535,536,537
D535  DEL=X
      GO TO 538
D537  DEL=X
      GO TO 539
 538  IF(NCT-12) 540,540,546
D540  ABDP=ABSF(PA-PP)
D     ABDP=ABDP/PP
      IF(ABDP-1.E-8) 536,536,541
D541  DEL=DEL/10.
      NCT=NCT+1
      IF(ABSF(DEL)-5.E-7*X) 536,536,542
D542  X=HX+DEL
D     HX=X
      INDEX=2
      GO TO 527
 533  IF(PA-PP) 542,536,539
 539  IF(NCT-12) 543,543,546
D543  ABDP=ABSF(PA-PP)
D     ABDP=ABDP/PP
      IF(ABDP-1.E-8) 536,536,544
D544  DEL=DEL/10.
      NCT=NCT+1
      IF(ABSF(DEL)-5.E-7*X) 536,536,545
D545  X=HX-DEL
D     HX=X
      IF(X) 563,563,564
D563  X=0.
 564  CONTINUE
      INDEX=3
      GO TO 527
 534  IF(PA-PP) 538,536,545
 536  AX(KA)=X
      GO TO 300
 546  AX(KA)=X
      PRINT 524,K12,K14
 524  FORMAT(13H0 IN ELEMENT I4,I4,61H FULL CONVERGENCE WAS NOT ATTAINED
     1. MAY BE SOMEWHAT UNPRECISE)
      GO TO 300
```

```
 199 CONTINUE
299  PRINT 298,K12,K15
298  FORMAT(20H0 ARGUMENT NEGATIVE I4,I4)
297  AX(KA)=-0.
300  CONTINUE
     RETURN
     END
```

```
      SUBROUTINE NCBTX(XK,AK,BK,CK,PK,ZK)
      COMMON NRA,NCA,NRB,NCB,NRC,NCC,NRD,NCD,NRE,NCE
      DIMENSION XK(1),AK(1),BK(1),CK(1),PK(1),ZK(1)
      DO 89 K15=1,NCA
      DO 89 K12=1,NRA
      KA=(K15-1)*NRA+K12
D     FLAM=0.
D     FIORD=0.
D     FIRES=0.
D     YPD=0.
D     XXD=0.
D     AAD=0.
D     BBD=0.
D     X=0.
      FLAM=CK(KA)
D     FLAM=FLAM/2.
      IF(FLAM) 80,4,4
    4 XXD=XK(KA)
      AAD=AK(KA)
      BBD=BK(KA)
D     A1=AAD
D     B1=BBD
      IF(AAD) 80,80,9
    9 IF(BBD) 80,80,11
   11 IF(XXD) 80,13,12
   13 PK(KA)=0.
      GO TO 491
D  12 X=XXD
      IF(X-1.) 14,14,80
   14 CONTINUE
      IF(ABSF(A1-1.)-1.E-7) 121,121,122
  121 IF(ABSF(B1-1.)-1.E-7) 123,123,122
  123 IF(FLAM-1.E-7) 124,124,125
  124 PK(KA)=X
      ZK(KA)=1.
      GO TO 89
D 125 C1=EXPF(FLAM*X-FLAM)
D     RESULT=X*C1
      PK(KA)=RESULT
D     ORD=C1*(1.+FLAM*X)
      ZK(KA)=ORD
      GO TO 89
  122 LFLAM=FLAM
D     XFLAM=LFLAM
D     A1=A1+XFLAM-1.
      LX=LFLAM-1
      LLX=LX+1
      INDE=1
      GO TO 488
D 486 A1=AAD+XFLAM
      INDE=2
  487 LX=LLX-1
      LLX=LX
      IF(LX) 497,489,489
D 489 AA1=LX
```

```
D       A1=A1-1.
        GO TO 485
  488 LX=LX+1
D       AA1=LX
D       A1=A1+1.
D 485 P=0.
D       PP=0.
D       RESULT=0.
        SENSE LIGHT 0
D402  CLBETA=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)
        IF(CLBETA+80.) 406,406,407
D 406 CBETA=0.
        GO TO 408
 407  IF(A1-1.) 411,412,411
 411  IF(B1-1.) 408,412,408
D 412 CBETA=EXPF(CLBETA)
 408  IF(X) 80,323,5
   5  IF(X-1.) 6,326,80
   6  IF(X-1.E-14) 323,323,7
D 7   XHH=0.
D       XHH=1.-X
        IF(XHH-1.E-14) 326,326,8
   8  IF(ABSF(A1-1.)-1.E-8) 433,433,434
  433 IF(ABSF(B1-1.)-1.E-8) 435,435,436
D 435 P=X
        GO TO 87
D 436 C1=B1*LOGF(1.-X)
        IF(C1+45.) 422,422,437
D 437 RESULT=1.-EXPF(C1)
        GO TO 429
  434 IF(ABSF(B1-1.)-1.E-8) 438,438,439
D 438 C1=A1*LOGF(X)
        IF(C1+45.) 423,423,440
D 440 RESULT=EXPF(C1)
        GO TO 429
  439 IF(A1-1000.) 416,416,417
 417  XX=2.*A1*(1.-X)/X
        DF=2.*B1
        PRINT 1995,K12,K15
 1995 FORMAT(29H0 CHIX APPROXIMATION USED IN 219)
        CALL CHI9(XX,DF,PRO,OR,K12,K15)
D       RESULT=1.-PRO
        IF(RESULT-.99999999) 429,420,420
D420  RESULT=1.
        GO TO 429
 416  IF(B1-1000.) 418,419,419
 419  XX=2.*B1*X/(1.-X)
        DF=2.*A1
        PRINT 1994,K12,K15
 1994 FORMAT(29H0 CHIX APPROXIMATION USED IN 2.9)
        CALL CHI9(XX,DF,PRO,OR,K12,K15)
D       RESULT=PRO
        IF(RESULT-.99999999) 429,420,420
  418 CONTINUE
        IF(A1-1.) 457,457,458
```

```
  457 IF(B1-1.) 459,458,458
  458 CONTINUE
      IF(X-.50) 472,473,473
D 473 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      SENSE LIGHT 3
  472 CONTINUE
      IF(A1-1.) 452,453,453
D 452 A1=A1+1.
D     B1  1-1.
      SENSE LIGHT 1
  453 CONTINUE
   70 II=80
D     FN=A1+B1-1.
D     XX=A1-1.
D     FMOD=FN*X
D     FO=LOGF(X)/5.
D     FO=EXPF(FO)
D     FMO=FMOD*FO
      IF(XX-FMO+2.) 425,447,447
D 425 XX=FN-XX
D     X=1.-X
  442 SENSE LIGHT 2
  447 CONTINUE
D     AA=X/(1.-X)
D     XXI=II
D     SS=((FN-XXI-XX)*(XX+XXI))/((XX+2.*XXI-1.)*(XX+2.*XXI))
D     SS=SS*AA
  108 II=II-1
D     AI=II
D     D1=(AI*(FN+AI))/((XX+2.*AI+1.)*(XX+2.*AI))
D     DD=D1*AA
D     C1=((FN-AI-XX)*(XX+AI))/((XX+2.*AI-1.)*(XX+2.*AI))
D     CC=C1*AA
D     SS=CC/(1.+DD/(1.-SS))
      IF(II-1) 109,109,108
D 109 SS=1./(1.-SS)
D     C1=ZLOGGM(FN+1.)-ZLOGGM(XX+2.)-ZLOGGM(FN-XX)+(XX+1.)*LOGF(X)+(FN-
     1XX-1.)*LOGF(1.-X)
D     SSUM=LOGF(SS)
D     SSUM1=SSUM+C1
      IF(SSUM1+80.) 423,423,110
D 423 RESULT=0.
      GO TO 421
D 110 SUM=EXPF(SSUM1)
D     RESULT=SUM
      GO TO 421
D 323 RESULT=0.
D     P=RESULT
      GO TO 86
D 326 RESULT=1.
D     P=RESULT
D     ORD=0.
```

```
       GO TO 86
   421 IF(SENSE LIGHT 2) 426,428
D  426 X=1.-X
D      XX=FN-XX
D      T1=ZLOGGM(FN+1.)-ZLOGGM(XX+1.)-ZLOGGM(FN-XX+1.)+XX*LOGF(X)+(FN-XX)
      1*LOGF(1.-X)
       IF(T1+45.) 460,460,461
D  460 RESULT=1.-RESULT
       GO TO 428
D  461 RESULT=1.-RESULT-EXPF(T1)
   428 IF(SENSE LIGHT1) 454,429
D  454 B1=B1+1.
D      C1=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)+(A1-1.)*LOGF(X)+(B1-1.)*LOG
      1F(1.-X)-LOGF(A1+B1-1.)
D      A1=A1-1.
       IF(C1+45.) 429,429,456
D  456 RESULT=RESULT+EXPF(C1)
       GO TO 429
   459 IF(A1-B1) 466,466,467
D  466 Y=A1
D      A1=B1
D      B1=Y
D      X=1.-X
       SENSE LIGHT 1
   467 IF(X-.85) 468,469,469
D  469 Y=A1
D      A1=B1
D      B1=Y
D      X=1.-X
       IF(SENSE LIGHT 1) 468,470
   470 SENSE LIGHT 1
   468 CONTINUE
D      Z=1.
D      C2=A1+1.
D230   SUMG=0.
D      SUMG=CLBETA+A1*LOGF(X)-LOGF(A1)
       IF(SUMG+80.) 295,295,240
D  24C RESULT=RESULT+EXPF(SUMG)
D  245 C16=1.-B1
D      SUMH=0.
D      SUMH=CLBETA+LOGF(C16)+C2*LOGF(X)-LOGF(C2)
  250  IF(SUMH+45.) 295,295,260
D  260 RESULT=RESULT+EXPF(SUMH)
D      C17=A1+Z
D      C18=Z+1.-B1
D      C19=A1+Z+1.
D      C20=Z+1.
D      SUMH=SUMH+LOGF(X)+LOGF(C17)+LOGF(C18)-LOGF(C19)-LOGF(C20)
D      Z=Z+1.
       GO TO 250
   295 CONTINUE
       IF(SENSE LIGHT 1) 471,429
D  471 Y=A1
D      A1=B1
D      B1=Y
```

```
D        X=1.-X
D        RESULT=1.-RESULT
  429 CONTINUE
      IF(SENSE LIGHT 3) 474,475
D 474 Y=A1
D        A1=B1
D        B1=Y
D        X=1.-X
D        RESULT=1.-RESULT
  475 CONTINUE
D        RCOM=1.-RESULT
      IF(RCOM-1.E-12) 422,422,427
D 422 RESULT=1.
D 427 P=RESULT
   87 CONTINUE
D        ORD1=CLBETA+(A1-1.)*LOGF(X)+(B1-1.)*LOGF(1.-X)
      IF(ORD1+45.) 403,403,404
D 403 ORD=0.
      GO TO 86
  404 IF(ORD1-80.) 415,405,405
D 405 ORD=.99999999E30
  415 ORD=EXPF(ORD1)
   86 IF(FLAM) 80,542,542
  542 IF(FLAM-1.E-7) 543,543,544
  543 PK(KA)=P
      ZK(KA)=ORD
      GO TO 89
  544 CFLAM=1000.
      IF(FLAM-CFLAM) 547,548,548
  548 PRINT 549,K12,K15
  549 FORMAT(68H0 NORMAL APPROXIMATION WAS USED FOR POISSON PART OF NONC
     1ENTRAL BETA 2I9)
      AA=(XX/FLAM)**.33333333-1.+2./(9.*FLAM)
      BB=AA/SQRTF(2./(9.*FLAM))
      CALL NORM9(BB,CC,PP1)
      PP=PP1
      GO TO 401
D 547 C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
      IF(C1+45.) 550,550,551
D 550 PP=0.
      GO TO 401
D 551 PP=EXPF(C1)
  401 CONTINUE
D        SUBRES=P*PP
D        SUBORD=PP*ORD
D        FIORD=FIORD+SUBORD
D        FIRES=FIRES+SUBRES
      GO TO(494,496),INDE
  494 IF(SUBRES-1.E-12) 486,486,488
  496 IF(SUBRES-1.E-12) 497,497,487
  497 PK(KA)=FIRES
      IF(X-1.E-12) 491,491,492
  491 IF(AAD-1.) 493,484,495
  493 ZK(KA)=.99999999E30
      GO TO 89
```

```
484 ZK(KA)=EXPF(-FLAM)
    GO TO 89
495 ZK(KA)=0.
    GO TO 89
492 ZK(KA)=FIORD
    GO TO 89
80  PRINT 81,K12,K15
81  FORMAT(26H0 ARGUMENT NOT ADMISSIBLE 2I6)
    PK(KA)=-0.
    ZK(KA)=-0.
89  CONTINUE
    RETURN
    END
```

```
      SUBROUTINE NCBTP(PL,AL,BL,CL,XL)
      DIMENSION PL(1),AL(1),BL(1),CL(1),XL(1)
      COMMON NRA,NCA,NRB,NCB,NRC,NCC,NRD,NCD,NRE NCE
      DO 57 K15=1,NCA
      DO 57 K12=1,NRA
      KA=(K15-1)*NRA+K12
D     PPD=0.
D     AAD=0.
D     BBD=0.
D     YXD=0.
D     X=0.
D     XC=0.
D     XDD=0.
D     FLAM=0.
      PPD=PL(KA)
      AAD=AL(KA)
      BBD=BL(KA)
      FLAM=CL(KA)
D     FLAM=FLAM/2.
      IF(FLAM) 52,4,4
    4 IF(AAD) 52,52,11
   11 IF(BBD) 52,52,12
   12 IF(PPD) 52,15,16
   15 XL(KA)=0.
      GO TO 57
   16 IF(PPD-1.) 18,18,52
D  18 PPH=0.
D     PPH=1.-PPD
      IF(PPH-1.E-8) 14,14,19
   19 IF(PPD-1.E-30) 15,15,17
D  17 A1=AAD
D     B1=BBD
D     PP=PPD
   74 IF(PP) 599,501,5020
D501  AX=0.
      GO TO 600
 5020 IF(PP-1.) 502,14,599
  502 IF(ABSF(A1-1.)-1.E-7) 121,121,122
  121 IF(ABSF(B1-1.)-1.E-7) 123,123,122
  123 IF(FLAM-1.E-7) 124,124,125
  124 XL(KA)=PP
      GO TO 57
D 125 C1=PP*EXPF(FLAM)
D     X=1.+LOGF(PP)/FLAM
      IF(X) 129,129,128
D 129 X=ABSF(X)*FLAM
D 128 X1=X+(LOGF(C1)-FLAM*X-LOGF(X))/(FLAM+1./X)
      IF(X1) 130,130,131
D 130 X1=1.E-6
      GO TO 132
  131 IF(X1-1.) 132,133,133
D 133 X1=1.-1.E-6
  132 CONTINUE
      IF(ABSF(X1-X)-1.E-8) 126,126,127
D 127 X=X1
```

```
      GO TO 128
  126 XL(KA)=XI
      GO TO 57
  122 IF(PP-.95) 134,135,135
D 135 X=(A1-1.+3.*FLAM)/(A1+B1-2.+3.*FLAM)
      GO TO 136
D 134 X=(A1-1.+FLAM)/(A1+B1-2.+FLAM)
  136 CONTINUE
      IF(X-1.E-16) 514,514,515
D 514 X=1.E-13
      GO TO 554
 515  IF(X-1.) 554,555,555
D 555 X=1.-1.E-12
  554 N=25
      III=0
      II=0
 516  II=II+1
D527  RESULT=0.
D     A1=AAD
D     HX=X
D     FIORD=0.
D     FIRES=0.
      LFLAM=FLAM
D     XFLAM=LFLAM
D     A1=A1+XFLAM-1.
      LX=LFLAM-1
      LLX=LX+1
      INDE=1
      GO TO 488
D 486 A1=AAD+XFLAM
      INDE=2
  487 LX=LLX-1
      LLX=LX
      IF(LX) 497,489,489
D 489 AA1=LX
D     A1=A1-1.
      GO TO 485
  488 LX=LX+1
D     AA1=LX
D     A1=A1+1.
D 485 PPO=0.
D     RESULT=0.
D     PA=0.
      SENSE LIGHT 0
D     CLBETA=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)
      IF(CLBETA+80.) 406,406,407
 406  CBETA=0.
      GO TO 408
 407  IF(A1-1.) 411,412,411
  411 IF(B1-1.) 408,412,402
 412  CBETA=EXPF(CLBETA)
  402 IF(A1-1.) 510,505,510
  505 IF(B1-1.) 506,422,506
D 506 CONST=B1*LOGF(1.-X)
      IF(CONST+45.) 422,422,507
```

```
D 507 RESULT=1.-EXPF(CONST)
      GO TO 475
  510 IF(B1-1.) 402,511,408
D 511 CONST=A1*LOGF(X)
      IF(CONST+45.) 427,427,512
D 512 RESULT=EXPF(CONST)
      GO TO 475
 408  IF(X) 1999,323,5
  5   IF(X-1.) 6,326,1999
  6   IF(X-1.E-14) 323,323,7
D 7   XHH=0.
D     XHH=1.-X
      IF(XHH-1.E-14) 326,326,8
   8  IF(ABSF(A1-1.)-1.E-8) 433,433,434
  433 IF(ABSF(B1-1.)-1.E-8) 435,435,436
D 435 PA=X
D     ORD=1.
      GO TO 2000
D 436 C1=B1*LOGF(1.-X)
      IF(C1+45.) 422,422,437
D 437 RESULT=1.-EXPF(C1)
      GO TO 429
  434 IF(ABSF(B1-1.)-1.E-8) 438,438,439
D 438 C1=A1*LOGF(X)
      IF(C1+45.) 423,423,440
D 440 RESULT=EXPF(C1)
      GO TO 429
  429 IF(A1-1000.) 416,416,417
  417 XX=2.*A1*(1.-X)/X
      DF=2.*B1
      PRINT 1995,K12,K15
 1995 FORMAT(29H0 CHIX APPROXIMATION USED IN 2191
      CALL CHI9(XX,DF,PRO,OR,K12,K15)
D     RESULT=1.-PRO
      IF(RESULT-.99999999) 429,420,420
D420  RESULT=1.
      GO TO 429
 416  IF(B1-1000.) 418,419,419
 419  XX=2.*B1*X/(1.-X)
      DF=2.*A1
      PRINT 1994,K12,K15
 1994 FORMAT(29H0 CHIX APPROXIMATION USED IN 2191
      CALL CHI9(XX,DF,PRO,OR,K12,K15)
D     RESULT=PRO
      IF(RESULT-.99999999) 429,420,420
  418 CONTINUE
      IF(A1-1.) 457,457,458
  457 IF(B1-1.) 459,458,458
  458 CONTINUE
      IF(X-.50) 472,473,473
D 473 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      SENSE LIGHT 3
```

```
  472 CONTINUE
      IF(A1-1.) 452,453,453
D 452 A1=A1+1.
D     B1=B1-1.
      SENSE LIGHT 1
  453 CONTINUE
   70 II=80
D     FN=A1+B1-1.
D     XX=A1-1.
D     FMOD=FN*X
D     FO=LOGF(X)/5.
D     FO=EXPF(FO)
D     FMO=FMOD*FO
      IF(XX-FMO+2.) 425,447,447
D 425 XX=FN-XX
D     X=1.-X
  442 SENSE LIGHT 2
  447 CONTINUE
D     AA=X/(1.-X)
D     XXI=II
D     SS=((FN-XXI-XX)*(XX+XXI))/((XX+2.*XXI-1.)*(XX+2.*XXI))
D     SS=SS*AA
  108 II=II-1
D     AI=II
D     D1=(AI*(FN+AI))/((XX+2.*AI+1.)*(XX+2.*AI))
D     DD=D1*AA
D     C1=((FN-AI-XX)*(XX+AI))/((XX+2.*AI-1.)*(XX+2.*AI))
D     CC=C1*AA
D     SS=CC/(1.+DD/(1.-SS))
      IF(II-1) 109,109,108
D 109 SS=1./(1.-SS)
D     C1=ZLOGGM(FN+1.)-ZLOGGM(XX+2.)-ZLOGGM(FN-XX)+(XX+1.)*LOGF(X)+(FN-
     1XX-1.)*LOGF(1.-X)
D     SSUM=LOGF(SS)
D     SSUM1=SSUM+C1
      IF(SSUM1+80.) 423,423,110
D 423 RESULT=0.
      GO TO 421
D 110 SUM=EXPF(SSUM1)
D     RESULT=SUM
      GO TO 421
D 323 RESULT=0.
D     PA=RESULT
      IF(A1-1.) 403,404,405
D403  ORD=.99999999E30
      GO TO 2000
D 404 ORD=OBETA
      GO TO 2000
D405  ORD=0.
      GO TO 2000
D 326 RESULT=1.
D     PA=RESULT
      IF(B1-1.) 403,404,405
  421 IF(SENSE LIGHT 2) 426,428
D 426 X=1.-X
```

```
D        XX=FN-XX
D        T1=ZLOGGM(FN+1.)-ZLOGGM(XX+1.)-ZLOGGM(FN-XX+1.)+XX*LOGF(X)+(FN-XX)
       1*LOGF(1.-X)
         IF(T1+45.) 460,460,461
D  460   RESULT=1.-RESULT
         GO TO 428
D  461   RESULT=1.-RESULT-EXPF(T1)
   428   IF(SENSE LIGHT1) 454,429
D  454   B1=B1+1.
D        C1=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)+(A1-1.)*LOGF(X)+(B1-1.)*LOG
       1F(1.-X)-LOGF(A1+B1-1.)
D        A1=A1-1.
         IF(C1+45.) 429,429,456
D  456   RESULT=RESULT+EXPF(C1)
         GO TO 429
   459   IF(A1-B1) 466,466,467
D  466   Y=A1
D        A1=B1
D        B1=Y
D        X=1.-X
         SENSE LIGHT 1
   467   IF(X-.85) 468,469,469
D  469   Y=A1
D        A1=B1
D        B1=Y
D        X=1.-X
         IF(SENSE LIGHT 1) 468,470
   470   SENSE LIGHT 1
   468   CONTINUE
D        Z=1.
D        C2=A1+1.
D230     SUMG=0.
D        SUMG=CLBETA+A1*LOGF(X)-LOGF(A1)
         IF(SUMG+80.) 295,295,240
D  240   RESULT=RESULT+EXPF(SUMG)
D  245   C16=1.-B1
D        SUMH=0.
D        SUMH=CLBETA+LOGF(C16)+C2*LOGF(X)-LOGF(C2)
   250   IF(SUMH+45.) 295,295,260
D  260   RESULT=RESULT+EXPF(SUMH)
D        C17=A1+Z
D        C18=Z+1.-B1
D        C19=A1+Z+1.
D        C20=Z+1.
D        SUMH=SUMH+LOGF(X)+LOGF(C17)+LOGF(C18)-LOGF(C19)-LOGF(C20)
D        Z=Z+1.
         GO TO 250
   295   CONTINUE
         IF(SENSE LIGHT 1) 471,429
D  471   Y=A1
D        A1=B1
D        B1=Y
D        X=1.-X
D        RESULT=1.-RESULT
   429   CONTINUE
```

```
      IF(SENSE LIGHT 3) 474,475
D 474 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
D     RESULT=1.-RESULT
  475 CONTINUE
D     RCOM=1.-RESULT
      IF(RCOM-1.E-12) 422,422,427
D 422 RESULT=1.
D 427 PA=RESULT
D     ORD1=CLBETA+(A1-1.)*LOGF(X)+(B1-1.)*LOGF(1.-X)
      IF(ORD1+45.) 405,405,409
 409  IF(ORD1-80.) 415,403,403
D 415 ORD=EXPF(ORD1)
 2000 IF(FLAM-1.E-7) 443,443,444
  444 CFLAM=1000.
      IF(FLAM-CFLAM) 483,448,448
  448 AA=(AA1/FLAM)**.33333333-1.+2./(9.*FLAM)
      BB=AA/SQRTF(2./(9.*FLAM))
      CALL NORM9(BB,CC,PP1)
      PPO=PP1
      GO TO 401
D 483 C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
      IF(C1+45.) 450,450,451
D 450 PPO=0.
      GO TO 401
D 451 PPO=EXPF(C1)
D 401 SUBRES=PPO*PA
D     SUBORD=PPO*ORD
D     FIORD=FIORD+SUBORD
D     FIRES=FIRES+SUBRES
      GO TO(494,496),INDE
  494 IF(SUBRES-1.E-12) 486,486,488
  496 IF(SUBRES-1.E-12) 497,497,487
D 497 PA=FIRES
D     ORD=FIORD
  443 IF(I1-1) 547,547,548
 547  INDEX=1
 548  GO TO(532,533,534),INDEX
 532  IF(ORD) 522,522,529
 529  CONTINUE
D     XC=X-(PA-PP)/ORD
D     DEL=0.
D     ABDP=ABSF(PA-PP)
D     ABDP=ABDP/PP
      IF(ABDP-1.E-8) 518,518,520
D518  AX=XC
      GO TO 600
D520  XHH=0.
D     XHH=1.-XC
      IF(XC) 137,137,138
  137 III=III+1
D     XXI=III
D     X=(.01)**XXI
```

```
  138 CONTINUE
      IF(XHH-1.E-15) 557,557,556
  556 IF(XC-1.E-30) 558,558,559
D557  X=(X+1.)/2.
      GO TO 560
D558  X=X/2.
      GO TO 560
D559  X=XC
  560 INDEX=1
      IF(I1-N) 516,516,522
  522 NCT=0
D     HX=X
      IF(X-1.E-30) 550,550,551
D551  XHH=0.
D     XHH=1.-X
      IF(XHH-1.E-15) 553,553,552
D550  X=.01
      GO TO 552
D553  X=.99
  552 IF(PA-PP) 535,536,537
D535  DEL=1.-X
      GO TO 538
D537  DEL=X
      GO TO 539
  538 IF(NCT-12) 540,540,546
D540  ABDP=ABSF(PA-PP)
D     ABDP=ABDP/PP
      IF(ABDP-1.E-8) 536,536,541
D541  DEL=DEL/10.
      NCT=NCT+1
      IF(ABSF(DEL)-1.E-14) 536,536,542
D542  X=HX+DEL
D     HX=X
      IF(X-1.) 562,561,561
D561  X=1.
  562 CONTINUE
      INDEX=2
      GO TO 527
  533 IF(PA-PP) 542,536,539
  539 IF(NCT-12) 543,543,546
D543  ABDP=ABSF(PA-PP)
D     ABDP=ABDP/PP
      IF(ABDP-1.E-8) 536,536,544
D544  DEL=DEL/10.
      NCT=NCT+1
      IF(ABSF(DEL)-1.E-14) 536,536,545
D545  X=HX-DEL
D     HX=X
      IF(X) 563,563,564
D563  X=0.
  564 CONTINUE
      INDEX=3
      GO TO 527
  534 IF(PA-PP) 538,536,545
D536  AX=X
```

```
      GO TO 600
D546  AX=X
      PRINT 524,K12    5
 524  FORMAT(13HO IN ...CMENT I4,I4,61H FULL CONVERGENCE WAS NOT ATTAINE
     1. MAY BE SOMEWHAT UNPRECISE)
      GO TO 600
 599  PRINT 598,K12,K15
 598  FORMAT(20HO ARGUMENT NEGATIVE I4,I4)
      AX=-0.
      GO TO 53
 600  CONTINUE
D     YXD=AX
D     AHH=1.-YXD
      IF(AHH-1.E-30) 14,14,13
   14 XL(KA)=1.
      GO TO 57
   13 XL(KA)=AX
      GO TO 57
 1999 CONTINUE
 52   PRINT 51,K12,K15
  51  FORMAT(26HO ARGUMENT NOT ADMISSIBLE I4,I4)
  53  XL(KA)=-0.
  57  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NCFX(XE,AE,BE,CE,PE,ZE)
      COMMON NRA,NCA,NRB,NCB,NRC,NCC,NRD,NCD,NRE,NCE
      DIMENSION XE(1),AE(1),BE(1),CE(1),PE(1),ZE(1)
      DO 89 K15=1,NCA
      DO 89 K12=1,NRA
      KA=(K15-1)*NRA+K12
D     FLAM=0.
D     FIORD=0.
D     FIRES=0.
D     YPD=0.
D     XXD=0.
D     AAD=0.
D     BBD=0.
D     X=0.
      FLAM=CE(KA)
D     FLAM=FLAM/2.
      XXD=XE(KA)
      AAD=AE(KA)
      BBD=BE(KA)
      IF(AAD) 80,80,9
    9 IF(BBD) 80,80,11
D  11 YXD=(XXD*AAD/BBD)/(1.+XXD*AAD/BBD)
D     YAD=AAD/2.
D     YBD=BBD/2.
D     A1=YAD
D     B1=YBD
      IF(YXD) 80,13,12
   13 PE(KA)=0.
      GO TO 491
D  12 X=YXD
      IF(X-1.) 14,14,80
   14 CONTINUE
      IF(ABSF(A1-1.)-1.E-7) 121,121,122
  121 IF(ABSF(B1-1.)-1.E-7) 123,123,122
  123 IF(FLAM-1.E-7) 124,124,125
  124 PE(KA)=X
      ZE(KA)=1.
      GO TO 89
D 125 C1=EXPF(FLAM*X-FLAM)
D     RESULT=X*C1
      PE(KA)=RESULT
D     ORD=C1*(1.+FLAM*X)
      ZE(KA)=ORD
      GO TO 89
  122 LFLAM=FLAM
D     XFLAM=LFLAM
D     A1=A1+XFLAM-1.
      LX=LFLAM-1
      LLX=LX+1
      INDE=1
      GO TO 488
D 486 A1=YAD+XFLAM
      INDE=2
  487 LX=LLX-1
      LLX=LX
```

```
      IF(LX) 497,489,489
D 489 AA1=LX
D     A1=A1-1.
      GO TO 485
  488 LX=LX+1
D     AA1=LX
D     A1=A1+1.
D 485 P=0.
D     PP=0.
D     RESULT=0.
      SENSE LIGHT 0
D402  CLBETA=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)
      IF(CLBETA+80.) 406,406,407
D 406 CBETA=0.
      GO TO 408
  407 IF(A1-1.) 411,412,411
  411 IF(B1-1.) 408,412,408
D 412 CBETA=EXPF(CLBETA)
  408 IF(X) 80,323,5
    5 IF(X-1.) 6,326,80
    6 IF(X-1.E-14) 323,323,7
D 7   XHH=0.
D     XHH=1.-X
      IF(XHH-1.E-14) 326,326,8
    8 IF(ABSF(A1-1.)-1.E-8) 433,433,434
  433 IF(ABSF(B1-1.)-1.E-8) 435,435,436
D 435 P=X
      GO TO 87
D 436 C1=B1*LOGF(1.-X)
      IF(C1+45.) 422,422,437
D 437 RESULT=1.-EXPF(C1)
      GO TO 429
  434 IF(ABSF(B1-1.)-1.E-8) 438,438,439
D 438 C1=A1*LOGF(X)
      IF(C1+45.) 423,423,440
D 440 RESULT=EXPF(C1)
      GO TO 429
  439 IF(A1-1000.) 416,416,417
  417 XX=2.*A1*(1.-X)/X
      DF=2.*B1
      PRINT 1995,K12,K15
 1995 FORMAT(29H0 CHIX APPROXIMATION USED IN 219)
      CALL CHI9(XX,DF,PRO,OR,K12,K15)
D     RESULT=1.-PRO
      IF(RESULT-.99999999) 429,420,420
D420  RESULT=1.
      GO TO 429
  416 IF(B1-1000.) 418,419,419
  419 XX=2.*B1*X/(1.-X)
      DF=2.*A1
      PRINT 1994,K12,K15
 1994 FORMAT(29H0 CHIX APPROXIMATION USED IN 219)
      CALL CHI9(XX,DF,PRO,OR,K12,K15)
D     RESULT=PRO
      IF(RESULT-.99999999) 429,420,420
```

```
  418 CONTINUE
      IF(A1-1.) 457,457,458
  457 IF(B1-1.) 459,458,458
  458 CONTINUE
      IF(X-.50) 472,473,473
D 473 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      SENSE LIGHT 3
  472 CONTINUE
      IF(A1-1.) 452,453,453
D 452 A1=A1+1.
D     B1=B1-1.
      SENSE LIGHT 1
  453 CONTINUE
   70 I1=80
D     FN=A1+B1-1.
D     XX=A1-1.
D     FMOD=FN*X
D     FO=LOGF(X)/5.
D     FO=EXPF(FO)
D     FMO=FMOD*FO
      IF(XX-FMO+2.) 425,447,447
D 425 XX=FN-XX
D     X=1.-X
  442 SENSE LIGHT 2
  447 CONTINUE
D     AA=X/(1.-X)
D     XXI=II
D     SS=((FN-XXI-XX)*(XX+XXI))/((XX+2.*XXI-1.)*(XX+2.*XXI))
D     SS=SS*AA
  108 II=II-1
D     AI=II
D     D1=(AI*(FN+AI))/((XX+2.*AI+1.)*(XX+2.*AI))
D     DD=D1*AA
D     C1=((FN-AI-XX)*(XX+AI))/((XX+2.*AI-1.)*(XX+2.*AI))
D     CC=C1*AA
D     SS=CC/(1.+DD/(1.-SS))
      IF(II-1) 109,109,108
D 109 SS=1./(1.-SS)
D     C1=ZLOGGM(FN+1.)-ZLOGGM(XX+2.)-ZLOGGM(FN-XX)+(XX+1.)*LOGF(X)+(FN-
     1XX-1.)*LOGF(1.-X)
D     SSUM=LOGF(SS)
D     SSUM1=SSUM+C1
      IF(SSUM1+80.) 423,423,110
D 423 RESULT=0.
      GO TO 421
D 110 SUM=EXPF(SSUM1)
D     RESULT=SUM
      GO TO 421
D 323 RESULT=0.
D     P=RESULT
      GO TO 86
D 326 RESULT=1.
```

```
D       P=RESULT
D       ORD=0.
        GO TO 86
  471   IF(SENSE LIGHT 2) 426,428
D 426   X=1.-X
D       XX=FN-XX
D       T1=ZLOGGM(FN+1.)-ZLOGGM(XX+1.)-ZLOGGM(FN-XX+1.)+XX*LOGF(X)+(FN-XX)
       1*LOGF(1.-X)
        IF(T1+45.) 460,460,461
D 460   RESULT=1.-RESULT
        GO TO 428
D 461   RESULT=1.-RESULT-EXPF(T1)
  428   IF(SENSE LIGHT1) 454,429
D 454   B1=B1+1.
D       C1=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)+(A1-1.)*LOGF(X)+(B1-1.)*LOG
       1F(1.-X)-LOGF(A1+B1-1.)
D       A1=A1-1.
        IF(C1+45.) 429,429,456
D 456   RESULT=RESULT+EXPF(C1)
        GO TO 429
  459   IF(A1-B1) 466,466,467
D 466   Y=A1
D       A1=B1
D       B1=Y
D       X=1.-X
        SENSE LIGHT 1
  467   IF(X-.85) 468,469,469
D 469   Y=A1
D       A1=B1
D       B1=Y
D       X=1.-X
        IF(SENSE LIGHT 1) 468,470
  470   SENSE LIGHT 1
  468   CONTINUE
D       Z=1.
D       C2=A1+1.
D230    SUMG=0.
D       SUMG=CLBETA+A1*LOGF(X)-LOGF(A1)
        IF(SUMG+80.) 295,295,240
D 240   RESULT=RESULT+EXPF(SUMG)
D 245   C16=1.-B1
D       SUMH=0.
D       SUMH=CLBETA+LOGF(C16)+C2*LOGF(X)-LOGF(C2)
  250   IF(SUMH+45.) 295,295,260
D 260   RESULT=RESULT+EXPF(SUMH)
D       C17=A1+Z
D       C18=Z+1.-B1
D       C19=A1+Z+1.
D       C20=Z+1.
D       SUMH=SUMH+LOGF(X)+LOGF(C17)+LOGF(C18)-LOGF(C19)-LOGF(C20)
D       Z=Z+1.
        GO TO 250
  295   CONTINUE
        IF(SENSE LIGHT 1) 471,429
D 471   Y=A1
```

```
D       A1=B1
D       B1=Y
D       X=1.-X
D       RESULT=1.-RESULT
  429  CONTINUE
       IF(SENSE LIGHT 3) 474,475
D 474  Y=A1
D       A1=B1
D       B1=Y
D       X=1.-X
D       RESULT=1.-RESULT
  475  CONTINUE
D       RCOM=1.-RESULT
       IF(RCOM-1.E-12) 422,422,427
D 422  RESULT=1.
D 427  P=RESULT
   87  CONTINUE
D       ORD1=CLBETA+(A1-1.)*LOGF(X)+(B1+1.)*LOGF(1.-X)+LOGF(YAD/YBD)
       IF(ORD1+45.) 403,403,404
D 403  ORD=0.
       GO TO 86
  404  IF(ORD1-80.) 415,405,405
D 405  ORD=.99999999E30
D 415  ORD=EXPF(ORD1)
   86  IF(FLAM) 80,542,542
  542  IF(FLAM-1.E-7) 543,543,544
  543  PE(KA)=P
       GO TO 89
  544  CFLAM=1000.
       IF(FLAM-CFLAM) 547,548,548
  548  PRINT 549,K12,K15
  549  FORMAT(65H0 NORMAL APPROXIMATION WAS USED FOR POISSON PART OF NONC
      1ENTRAL F 2I9)
       AA=(XX/FLAM)**.33333333-1.+2./(9.*FLAM)
       BB=AA/SQRTF(2./(9.*FLAM))
       CALL NORM9(BB,CC,PP1)
       PP=PP1
       GO TO 401
D 547  C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
       IF(C1+45.) 550,550,551
D 550  PP=0.
       GO TO 401
D 551  PP=EXPF(C1)
  401  CONTINUE
D       SUBRES=P*PP
D       SUBORD=PP*ORD
D       FIORD=FIORD+SUBORD
D       FIRES=FIRES+SUBRES
       GO TO(494,496),INDE
  494  IF(SUBRES-1.E-12) 486,486,488
  496  IF(SUBRES-1.E-12) 497,497,487
  497  PE(KA)=FIRES
       IF(YXD-1.E-12) 491,491,492
  491  IF(YAD-1.) 493,484,495
  493  ZE(KA)=.99999999E30
```

```
      GO TO 89
  -84 ZE(KA)=EXPF(-FLAM)
      GO TO 89
  495 ZE(KA)=0.
      GO TO 89
  492 ZE(KA)=FIORD
      GO TO 89
  80  PRINT 81,K12,K15
  81  FORMAT(26H0 ARGUMENT NOT ADMISSIBLE 2I6)
      PE(KA)=-0.
  89  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NCFP(PH,AH,BH,CH,XH)
      DIMENSION PH(1),AH(1),BH(1),CH(1),XH(1)
      COMMON NRA,NCA,NRB,NCB,NRC,NCC,NRD,NCD,NRE,NCE
      DO 57 K15=1,NCA
      DO 57 K12=1,NRA
      KA=(K15-1)*NRA+K12
D     PPD=0.
D     AAD=0.
D     BBD=0.
D     YXD=0.
D     X=0.
D     XC=0.
D     XDD=0.
D     FLAM=0.
      PPD=PH(KA)
      AAD=AH(KA)
      BBD=BH(KA)
      FLAM-CH(KA)
D     FLAM=FLAM/2.
      IF(AAD) 52,52,11
 11   IF(BBD) 52,52,12
 12   IF(PPD) 52,15,16
   15 XH(KA)=0.
       GO TO 57
  16  IF(PPD-1.) 18,18,52
D 18  PPH=0.
D     PPH=1.-PPD
      IF(PPH-1.E-8) 14,14,19
  19  IF(PPD-1.E-30) 15,15,17
D 17  YAD=AAD/2.
D     YBD=BBD/2.
D     YPD=PPD
D     A1=YAD
D     B1=YBD
D     PP=YPD
  74  IF(PP) 599,501,5020
D501  AX=0.
      GO TO 600
 5020 IF(PP-1.) 502,14,599
  502 IF(ABSF(A1-1.)-1.E-7) 121,121,122
  121 IF(ABSF(B1-1.)-1.E-7) 123,123,122
  123 IF(FLAM-1.E-7) 124,124,125
D 124 AX=PP
      GO TO 600
D 125 C1=PP*EXPF(FLAM)
D     X=1.+LOGF(PP)/FLAM
      IF(X) 129,129,128
D 129 X=ABSF(X)*FLAM
D 128 X1=X+(LOGF(C1)-FLAM*X-LOGF(X))/(FLAM+1./X)
      IF(X1) 130,130,131
D 130 X1=1.E-6
      GO TO 132
  131 IF(X1-1.) 132,133,133
D 133 X1=1.-1.E-6
  132 CONTINUE
```

```
      IF(ABSF(X1-X)-1.E-8) 126,126,127
D 127 X=X1
      GO TO 128
D 126 AX=X1
      GO TO 600
  122 IF(PP-.95) 134,135,135
D 135 X=(A1-1.+3.*FLAM)/(A1+B1-2.+3.*FLAM)
      GO TO 136
D 134 X=(A1-1.+FLAM)/(A1+B1-2.+FLAM)
  136 CONTINUE
      IF(X-1.E-16) 514,514,515
D 514 X=1.E-13
      GO TO 554
  515 IF(X-1.) 554,555,555
D 555 X=1.-1.E-12
  554 N=25
      III=0
      I1=0
  516 I1=I1+1
D527  RESULT=0.
D     HX=X
D     FIORD=0.
D     FIRES=0.
      LFLAM=FLAM
D     XFLAM=LFLAM
D     A1=A1+XFLAM-1.
      LX=LFLAM-1
      LLX=LX+1
      INDE=1
      GO TO 488
D 486 A1=AO+XFLAM
      INDE=2
  487 LX=LLX-1
      LLX=LX
      IF(LX) 497,489,489
D 489 AA1=LX
D     A1=A1-1.
      GO TO 485
  488 LX=LX+1
C     AA1=LX
D     A1=A1+1.
D 485 PPO=0.
D     PESULT=0.
D     PA=0.
      SENSE LIGHT 0
D     CLBETA=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)
      IF(CLBETA+80.) 406,406,407
  406 CBETA=0.
      GO TO 408
  407 IF(A1-1.) 411,412,411
  411 IF(B1-1.) 408,412,402
  412 CBETA=EXPF(CLBETA)
  402 IF(A1-1.) 510,505,510
  505 IF(B1-1.) 506,422,506
D 506 CONST=B1*LOGF(1.-X)
```

```
      IF(CONST+45.) 422,422,507
D 507 RESULT=1.-EXPF(CONST)
      GO TO 475
  510 IF(B1-1.) 402,511,408
D 511 CONST=A1*LOGF(X)
      IF(CONST+45.) 427,427,512
D 512 RESULT=EXPF(CONST)
      GO TO 475
  408 IF(X) 1999,323,5
    5 IF(X-1.) 6,326,1999
    6 IF(X-1.E-14) 323,323,7
D   7 XHH=0.
D     XHH=1.-X
      IF(XHH-1.E-14) 326,326,8
    8 IF(ABSF(A1-1.)-1.E-8) 433,433,434
  433 IF(ABSF(B1-1.)-1.E-8) 435,435,436
D 435 PA=X
D     ORD=1.
      GO TO 2000
D 436 C1=B1*LOGF(1.-X)
      IF(C1+45.) 422,422,437
D 437 RESULT=1.-EXPF(C1)
      GO TO 429
  434 IF(ABSF(B1-1.)-1.E-8) 438,438,439
D 438 C1=A1*LOGF(X)
      IF(C1+45.) 423,423,440
D 440 RESULT=EXPF(C1)
      GO TO 429
  439 IF(A1-1000.) 416,416,417
  417 XX=2.*A1*(1.-X)/X
      DF=2.*B1
      PRINT 1995,K12,K15
 1995 FORMAT(29H0 CHIX APPROXIMATION USED IN 2I9)
      CALL CHI9(XX,DF,PRO,OR,K12,K15)
D     RESULT=1.-PRO
      IF(RESULT-.99999999) 429,420,420
D420  RESULT=1.
      GO TO 429
  416 IF(B1-1000.) 418,419,419
  419 XX=2.*B1*X/(1.-X)
      DF=2.*A1
      PRINT 1994,K12,K15
 1994 FORMAT(29H0 CHIX APPROXIMATION USED IN 2I9)
      CALL CHI9(XX,DF,PRO,OR,K12,K15)
D     RESULT=PRO
      IF(RESULT-.99999999) 429,420,420
  418 CONTINUE
      IF(A1-1.) 457,457,458
  457 IF(B1-1.) 459,458,458
  458 CONTINUE
      IF(X-.50) 472,473,473
D 473 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
```

```
      SENSE LIGHT 3
  472 CONTINUE
      IF(A1-1.) 452,453,453
D 452 A1=A1+1.
D     B1=B1-1.
      SENSE LIGHT 1
  453 CONTINUE
   73 II=80
D     FN=A1+B1-1.
D     XX=A1-1.
D     FMOD=FN*X
D     FO=LOGF(X)/5.
D     FO=EXPF(FO)
D     FMO=FMOD*FO
      IF(XX-FMO+2.) 425,447,447
D 425 XX=FN-XX
D     X=1.-X
  442 SENSE LIGHT 2
  447 CONTINUE
D     AA=X/(1.-X)
D     XXI=II
D     SS=((FN-XXI-XX)*(XX+XXI))/((XX+2.*XXI-1.)*(XX+2.*XXI))
D     SS=SS*AA
  108 II=II-1
D     AI=II
D     D1=(AI*(FN+AI))/((XX+2.*AI+1.)*(XX+2.*AI))
D     DD=D1*AA
D     C1=((FN-AI-XX)*(XX+AI))/((XX+2.*AI-1.)*(XX+2.*AI))
D     CC=C1*AA
D     SS=CC/(1.+DD/(1.-SS))
      IF(II-1) 109,109,108
D 109 SS=1./(1.-SS)
D     C1=ZLOGGM(FN+1.)-ZLOGGM(XX+2.)-ZLOGGM(FN-XX)+(XX+1.)*LOGF(X)+(FN-
     1XX-1.)*LOGF(1.-X)
D     SSUM=LOGF(SS)
D     SSUM1=SSUM+C1
      IF(SSUM1+80.) 423,423,110
D 423 RESULT=0.
      GO TO 421
D 110 SUM=EXPF(SSUM1)
D     RESULT=SUM
      GO TO 421
D 323 RESULT=0.
D     PA=RESULT
      IF(A1-1.) 403,404,405
D403  ORD=.99999999E30
      GO TO 2000
D 404 ORD=CBETA
      GO TO 2000
D405  ORD=0.
      GO TO 2000
D 326 RESULT=1.
D     PA=RESULT
      IF(B1-1.) 403,404,405
  421 IF(SENSE LIGHT 2) 426,428
```

```
D 426 X=1.-X
D     XX=FN-XX
D     T1=ZLOGGM(FN+1.)-ZLOGGM(XX+1.)-ZLOGGM(FN-XX+1.)+XX*LOGF(X)+(FN-XX)
     1*LOGF(1.-X)
      IF(T1+45.) 460,460,461
D 460 RESULT=1.-RESULT
      GO TO 428
D 461 RESULT=1.-RESULT-EXPF(T1)
  428 IF(SENSE LIGHT1) 454,429
D 454 B1=B1+1.
D     C1=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)+(A1-1.)*LOGF(X)+(B1-1.)*LOG
     1F(1.-X)-LOGF(A1+B1-1.)
D     A1=A1-1.
      IF(C1+45.) 429,429,456
D 456 RESULT=RESULT+EXPF(C1)
      GO TO 429
  459 IF(A1-B1) 466,466,467
D 466 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      SENSE LIGHT 1
  467 IF(X-.85) 468,469,469
D 469 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      IF(SENSE LIGHT 1) 468,470
  470 SENSE LIGHT 1
  468 CONTINUE
D     Z=1.
D     C2=A1+1.
D230  SUMG=0.
D     SUMG=CLBETA+A1*LOGF(X)-LOGF(A1)
      IF(SUMG+80.) 295,295,240
D 240 RESULT=RESULT+EXPF(SUMG)
D 245 C16=1.-B1
D     SUMH=0.
D     SUMH=CLBETA+LOGF(C16)+C2*LOGF(X)-LOGF(C2)
  250 IF(SUMH+45.) 295,295,260
D 260 RESULT=RESULT+EXPF(SUMH)
D     C17=A1+Z
D     C18=Z+1.-B1
D     C19=A1+Z+1.
D     C20=Z+1.
D     SUMH=SUMH+LOGF(X)+LOGF(C17)+LOGF(C18)-LOGF(C19)-LOGF(C20)
D     Z=Z+1.
      GO TO 250
  295 CONTINUE
      IF(SENSE LIGHT 1) 471,429
D 471 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
D     RESULT=1.-RESULT
```

```
  429 CONTINUE
      IF(SENSE LIGHT 3) 474,475
D 474 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
D     RESULT=1.-RESULT
  475 CONTINUE
D     RCOM=1.-RESULT
      IF(RCOM-1.E-12) 422,422,427
D 422 RESULT=1.
D 427 PA=RESULT
D     ORD1=CLBETA+(A1-1.)*LOGF(X)+(B1-1.)*LOGF(1.-X)
      IF(ORD1+45.) 405,405,409
  409 IF(ORD1-80.) 415,403,403
D 415 ORD=EXPF(ORD1)
 2000 IF(FLAM-1.E-7) 443,443,444
  444 CFLAM=1000.
      IF(FLAM-CFLAM) 483,448,448
  448 AA=(AA1/FLAM)**.33333333-1.+2./(9.*FLAM)
      BB=AA/SQRTF(2./(9.*FLAM))
      CALL NORM9(BB,CC,PP1)
      PPO=PP1
      GO TO 401
D 483 C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
      IF(C1+45.) 450,450,451
D 450 PPO=0.
      GO TO 401
D 451 PPO=EXPF(C1)
D 401 SUBRES=PPO*PA
D     SUBORD=PPO*ORD
D     FIORD=FIORD+SUBORD
D     FIRES=FIRES+SUBRES
      GO TO(494,496),INDE
  494 IF(SUBRES-1.E-12) 486,486,488
  496 IF(SUBRES-1.E-12) 497,497,487
D 497 PA=FIRES
D     ORD=FIORD
  443 IF(I1-1) 547,547,548
  547 INDEX=1
  548 GO TO(532,533,534),INDEX
  532 IF(ORD) 522,522,529
  529 CONTINUE
D     XC=X-(PA-PP)/ORD
D     DEL=0.
D     ABDP=ABSF(PA-PP)
D     ABDP=ABDP/PP
      IF(ABDP-1.E-8) 518,518,520
D518  AX=XC
      GO TO 600
D520  XHH=0.
D     XHH=1.-XC
      IF(XC) 137,137,138
  137 III=III+1
D     XXI=III
```

```
D       X=(.01)**XXI
   138  CONTINUE
        IF(XHH-1.E-15) 557,557,556
  556   IF(XC-1.E-30) 558,558,559
 D557   X=(X+1.)/2.
        GO TO 560
 D558   X=X/2.
        GO TO 560
 D559   X=XC
  560   INDEX=1
        IF(I1-N) 516,516,522
   522  NCT=0
 D      HX=X
        IF(X-1.E-30) 550,550,551
 D551   XHH=0.
 D      XHH=1.-X
        IF(XHH-1.E-15) 553,553,552
 D550   X=.01
        GO TO 552
 D553   X=.99
  552   IF(PA-PP) 535,536,537
 D535   DEL=1.-X
        GO TO 538
 D537   DEL=X
        GO TO 539
  538   IF(NCT-12) 540,540,546
 D540   ABDP=ABSF(PA-PP)
 D      ABDP=ABDP/PP
        IF(ABDP-1.E-8) 536,536,541
 D541   DEL=DEL/10.
        NCT=NCT+1
        IF(ABSF(DEL)-1.E-14) 536,536,542
 D542   X=HX+DEL
 D      HX=X
        IF(X-1.) 562,561,561
 D561   X=1.
  562   CONTINUE
        INDEX=2
        GO TO 527
  533   IF(PA-PP) 542,536,539
  539   IF(NCT-12) 543,543,546
 D543   ABDP=ABSF(PA-PP)
 D      ABDP=ABDP/PP
        IF(ABDP-1.E-8) 536,536,544
 D544   DEL=DEL/10.
        NCT=NCT+1
        IF(ABSF(DEL)-1.E-14) 536,536,545
 D545   X=HX-DEL
 D      HX=X
        IF(X) 563,563,564
 D563   X=0.
  564   CONTINUE
        INDEX=3
        GO TO 527
  534   IF(PA-PP) 538,536,545
```

```
D536  AX=X
      GO TO 600
D546  AX=X
      PRINT 524,K12,K15
 524  FORMAT(13H0 IN ELEMENT I4,I4,61H FULL CONVERGENCE WAS NOT ATTAINED
     1. MAY BE SOMEWHAT UNPRECISE)
      GO TO 600
 599  PRINT 598,K12,K15
 598  FORMAT(20H0 ARGUMENT NEGATIVE I4,I4)
      AX=-0.
      GO TO 53
 600  CONTINUE
D     YXD=AX
D     AHH=1.-YXD
      IF(AHH-1.E-30) 14,14,13
   14 XH(KA)=.99999999E30
      GO TO 57
D 13  XDD=(YXD*BBD)/(AAD*(1.-YXD))
      XH(KA)=XDD
      GO TO 57
 1999 CONTINUE
 52   PRINT 51,K12,K15
  51  FORMAT(26H0 ARGUMENT NOT ADMISSIBLE I4,I4)
   53 XH(KA)=-0.
  57  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NCTX(TE,AE,CE,PE,ORE)
      COMMON NRA,NCA,NRB,NCB,NRC,NCC,NRD,NCD
      DIMENSION TE(1),AE(1),CE(1),PE(1),ORE(1)
      DO 89 K15=1,NCA
      DO 89 K12=1,NRA
      KA=(K15-1)*NRA+K12
D     POSUM=0.
D     FLAM=0.
D     FIRES=0.
D     FIORD=0.
D     YPD=0.
D     XXD=0.
D     AAD=0.
D     BBD=0.
D     X=0.
      FLAM=CE(KA)
D     FLAM=FLAM/2.
      XXD=TE(KA)
      AAD=AE(KA)
      IF(AAD) 80,80,11
   11 IF(FLAM) 80,12,12
D  12 YXD=(XXD**2)/AAD
D     B1=AAD/2.
D     X=YXD/(1.+YXD)
      IF(FLAM-15.) 462,462,463
D 462 A1A=0.
      INDE=5
      GO TO 488
D 464 A1=.5
      INDE=6
      GO TO 485
  463 LAM=FLAM
D     XLAM=LAM
D     A1A=XLAM-1.
      INDE=1
      GO TO 488
D 487 A1A=XLAM
      INDE=2
      GO TO 486
D 489 A1A=XLAM-.5
      INDE=3
      GO TO 488
D 499 A1A=XLAM+.5
      INDE=4
      GO TO 486
D 488 A1=A1A+1.
      GO TO 485
D 486 A1=A1A-1.
D 485 A1A=A1
D     AA1=A1A-.5
D     P=0.
D     PP=0.
D     RESULT=0.
      SENSE LIGHT 0
D402  CLBETA=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)
```

```
      IF(CLBETA+80.) 406,406,407
D 406 CBETA=0.
      GO TO 408
 407  IF(A1-1.) 411,412,411
 411  IF(B1-1.) 408,412,408
D 412 CBETA=EXPF(CLBETA)
  408 IF(X) 80,323,5
    5 IF(X-1.E-14) 323,323,7
D 7   XHH=0.
D     XHH=1.-X
      IF(XHH-1.E-14) 326,326,8
    8 IF(ABSF(A1-1.)-1.E-8) 433,433,434
  433 IF(ABSF(B1-1.)-1.E-8) 435,435,436
D 435 P=X
      GO TO 87
D 436 C1=B1*LOGF(1.-X)
      IF(C1+45.) 422,422,437
D 437 RESULT=1.-EXPF(C1)
      GO TO 429
  434 IF(ABSF(B1-1.)-1.E-8) 438,438,439
D 438 C1=A1*LOGF(X)
      IF(C1+45.) 423,423,440
D 440 RESULT=EXPF(C1)
      GO TO 429
  439 IF(A1-1000.) 416,416,417
 417  XX=2.*A1*(1.-X)/X
      DF=2.*B1
      PRINT 1995,K12,K15
 1995 FORMAT(29H0 CHIX APPROXIMATION USED IN 219)
      CALL CHI9(XX,DF,PRO,OR,K12,K15)
D     RESULT=1.-PRO
      IF(RESULT-.99999999) 429,420,420
D420  RESULT=1.
      GO TO 429
 416  IF(B1-1000.) 418,419,419
 419  XX=2.*B1*X/(1.-X)
      DF=2.*A1
      PRINT 1994,K12,K15
 1994 FORMAT(29H0 CHIX APPROXIMATION USED IN 219)
      CALL CHI9(XX,DF,PRO,OR,K12,K15)
D     RESULT=PRO
      IF(RESULT-.99999999) 429,420,420
  418 CONTINUE
      IF(A1-1.) 457,457,458
  457 IF(B1-1.) 459,458,458
  458 CONTINUE
      IF(X-.50) 472,473,473
D 473 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      SENSE LIGHT 3
  472 CONTINUE
      IF(A1-1.) 452,453,453
D 452 A1=A1+1.
```

```
D      B1=B1-1.
       SENSE LIGHT 1
  453  CONTINUE
   70  II=80
D      FN=AI+B1-1.
D      XX=AI-1.
D      FMOD=FN*X
D      FO=LOGF(X)/5.
D      FO=EXPF(FO)
D      FMO=FMOD*FO
       IF(XX-FMO+2.) 425,447,447
D 425  XX=FN-XX
D      X=1.-X
  442  SENSE LIGHT 2
  447  CONTINUE
D      AA=X/(1.-X)
D      XXI=II
D      SS=((FN-XXI-XX)*(XX+XXI))/((XX+2.*XXI-1.)*(XX+2.*XXI))
D      SS=SS*AA
  108  II=II-1
D      AI=II
D      D1=(AI*(FN+AI))/((XX+2.*AI+1.)*(XX+2.*AI))
D      DD=D1*AA
D      C1=((FN-AI-XX)*(XX+AI))/((XX+2.*AI-1.)*(XX+2.*AI))
D      CC=C1*AA
D      SS=CC/(1.+DD/(1.-SS))
       IF(II-1) 109,109,108
D 109  SS=1./(1.-SS)
D      C1=ZLOGGM(FN+1.)-ZLOGGM(XX+2.)-ZLOGGM(FN-XX)+(XX+1.)*LOGF(X)+(FN-
      1XX-1.)*LOGF(1.-X)
D      SSUM=LOGF(SS)
D      SSUM1=SSUM+C1
       IF(SSUM1+80.) 423,423,110
D 423  RESULT=0.
       GO TO 421
D 110  SUM=EXPF(SSUM1)
D      RESULT=SUM
       GO TO 421
D 323  RESULT=0.
D      P=RESULT
       GO TO 482
D 326  RESULT=1.
D      P=RESULT
D      ORD=0.
       GO TO 482
  421  IF(SENSE LIGHT 2) 426,428
D 426  X=1.-X
D      XX=FN-XX
D      T1=ZLOGGM(FN+1.)-ZLOGGM(XX+1.)-ZLOGGM(FN-XX+1.)+XX*LOGF(X)+(FN-XX)
      1*LOGF(1.-X)
       IF(T1+45.) 460,460,461
D 460  RESULT=1.-RESULT
       GO TO 428
D 461  RESULT=1.-RESULT-EXPF(T1)
  428  IF(SENSE LIGHT1) 454,429
```

```
D 454 A1=B1+1.
D     C1=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)+(A1-1.)*LOGF(X)+(B1-1.)*
     1F(1.-X)-LOGF(A1+B1-1.)
D     A1=A1-1.
      IF(C1+45.) 429,429,456
D 456 RESULT=RESULT+EXPF(C1)
      GO TO 429
  459 IF(A1-B1) 466,466,467
D 466 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      SENSE LIGHT 1
  467 IF(X-.85) 468,469,469
D 469 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      IF(SENSE LIGHT 1) 468,470
  470 SENSE LIGHT 1
  468 CONTINUE
D     Z=1.
D     C2=A1+1.
D230  SUMG=0.
D     SUMG=CLBETA+A1*LOGF(X)-LOGF(A1)
      IF(SUMG+80.) 295,295,240
D 240 RESULT=RESULT+EXPF(SUMG)
D 245 C16=1.-B1
D     SUMH=0.
D     SUMH=CLBETA+LOGF(C16)+C2*LOGF(X)-LOGF(C2)
  250 IF(SUMH+45.) 295,295,260
D 260 RESULT=RESULT+EXPF(SUMH)
D     C17=A1+Z
D     C18=Z+1.-B1
D     C19=A1+Z+1.
D     C20=Z+1.
D     SUMH=SUMH+LOGF(X)+LOGF(C17)+LOGF(C18)-LOGF(C19)-LOGF(C20)
D     Z=Z+1.
      GO TO 250
  295 CONTINUE
      IF(SENSE LIGHT 1) 471,429
D 471 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
D     RESULT=1.-RESULT
  429 CONTINUE
      IF(SENSE LIGHT 3) 474,475
D 474 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
D     RESULT=1.-RESULT
  475 CONTINUE
D     RCOM=1.-RESULT
```

```
      IF(RCOM-1.E-12) 422,422,427
D 422 RESULT=1.
D 427 P=RESULT
   87 CONTINUE
D     ORD1=CLBETA+(A1-.5)*LOGF(X)+(B1+.5)*LOGF(1.-X)-.5*LOGF(AAD)
      IF(ORD1+45.) 101,101,102
D 101 ORD=0.
      GO TO 482
  102 IF(ORD1-80.) 415,103,103
D 103 ORD=.99999999E30
      GO TO 482
D 415 ORD=EXPF(ORD1)
  482 IF(FLAM-1.E-7) 443,443,444
 443  PE(KA)=P
      ORE(KA)=ORD
      GO TO 89
 444  CFLAM=1000.
      IF(FLAM-CFLAM) 483,448,448
 448  PRINT 449,K12,K15
 449  FORMAT(65H0 NORMAL APPROXIMATION WAS USED FOR POISSON PART OF NONC
     1ENTRAL F 2I9)
     AA=(AA1/FLAM)**.33333333-1.+2./(9.*FLAM)
     BB=AA/SQRTF(2./(9.*FLAM))
     CALL NORM9(BB,CC,PP1)
     PP=PP1
      GO TO 401
D 483 C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
      IF(C1+45.) 450,450,451
D450  PP=0.
      GO TO 401
D451  PP=EXPF(C1)
 401  CONTINUE
D     POSUM=POSUM+PP
D     SUBRES=P*PP
D     SUBORD=PP*ORD
D     FIORD=FIORD+SUBORD
D     FIRES=FIRES+SUBRES
      GO TO(494,497,494,496,476,476),INDE
  476 IF(PP-1.E-12) 484,484,488
  494 IF(PP-1.E-5) 484,484,488
  497 IF(A1A-1.9) 484,484,498
  498 IF(PP-1.E-12) 484,484,486
  496 IF(A1A-1.4) 484,484,498
  484 GO TO(487,490,499,491,490,491),INDE
D 490 FRES1=FIRES
D     FORD1=FIORD
D     FPOSM=POSUM
D     FIRES=0.
D     FIORD=0.
D     POSUM=0.
      GO TO(487,489,499,491,464,491),INDE
  491 IF(XXD) 492,493,493
D 492 FRES=(1.-FPOSM+FRES1-FIRES)/2.
D     FORD=FIORD-FORD1
      GO TO 481
```

```
  493 IF(ABSF(XXD)-1.E-12) 403,403,404
D 403 FRES=(1.-FPOSM)/2.
D     ORD1=ZLOGGM(B1+.5)-ZLOGGM(.5)-ZLOGGM(B1)-FLAM-.5*LOGF(AAD)
      IF(ORD1+45.) 405,405,495
D 405 FORD=0.
      GO TO 481
D 495 FORD=EXPF(ORD1)
      GO TO 481
D 404 FRES=(1.-FPOSM+FRES1+FIRES)/2.
D     FORD=FIORD+FORD1
  481 CONTINUE
      IF(FRES) 115,115,116
D 115 FRES=0.
  116 CONTINUE
      PE(KA)=FRES
      IF(FORD-.99999999E30) 18,19,19
   19 ORE(KA)=.99999999E30
      GO TO 89
   18 ORE(KA)=FORD
      GO TO 89
   80 PRINT 81,K12,K15
   81 FORMAT(26H0 ARGUMENT NOT ADMISSIBLE 2I6)
      PE(KA)=-0.
      ORE(KA)=-0.
   89 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NCTP(PQ,AQ,CQ,XQ)
      DIMENSION PQ(1),AQ(1),CQ(1),XQ(1)
      COMMON NRA,NCA,NRB,NCB,NRC,NCC,NRD,NCD
      DO 57 K15=1,NCA
      DO 57 K12=1,NRA
      KA=(K15-1)*NRA+K12
D     PPD=0.
D     AAD=0.
D     YXD=0.
D     X=0.
D     XC=0.
D     XDD=0.
D     FLAM=0.
      PPD=PQ(KA)
      AAD=AQ(KA)
      FLAM=CQ(KA)
D     FLAM=FLAM/2.
      IF(AAD) 52,52,12
   12 IF(PPD) 52,15,16
   15 XQ(KA)=-.99999999E30
      GO TO 57
   16 IF(PPD-1.) 18,18,52
D  18 PPH=0.
D     PPH=1.-PPD
      IF(PPH-1.E-30) 14,14,19
   19 IF(PPD-1.E-30) 15,15,17
D  17 B1=AAD/2.
D     PP=PPD
D     PU=0.
      IF(FLAM-1.E-6) 126,126,127
  127 IF(FLAM-1000.) 128,129,129
D 129 PU=0.
      GO TO 137
D 126 PU=.5
      GO TO 137
  128 LAM=FLAM
D     XFLAM=LAM
D     AA1=XFLAM+.5
D 132 C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
      IF(C1+30.) 130,130,131
D 131 PU=PU+EXPF(C1)
D     AA1=AA1+1.
      GO TO 132
D 130 AA1=XFLAM-.5
  135 IF(AA1) 133,133,134
D 134 C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
      IF(C1+30.) 133,133,136
D 136 PU=PU+EXPF(C1)
D     AA1=AA1-1.
      GO TO 135
D 133 PU=.5-PU/2.
D 137 CPU=ABSF(PP-PU)
      IF(CPU-.05) 123,123,124
D 123 X=CPU
      GO TO 125
```

```
  124 IF(PP-.01) 138,138,139
  139 IF(PP-.99) 140,138,139
D 138 X=1.-PP
      GO TO 125
D 140 X=PP
  125 CONTINUE
      IF(X-1.E-30) 514,514,515
D 514 X=1.E-16
      GO TO 554
 515  IF(X-1.) 554,555,555
D 555 X=1.-1.E-12
  554 N=30
      I1=0
 516  I1=I1+1
D527  RESULT=0.
D     HX=X
D     POSUM=0.
D     FIORD=0.
D     FIRES=0.
      IF(FLAM-15.) 462,462,463
D 462 A1A=0.
      INDE=5
      GO TO 498
D 464 A1=.3
      INDE=6
      GO TO 485
  463 LAM=FLAM
D     XLAM=LAM
D     A1A=XLAM-1.
      INDE=1
      GO TO 488
D 487 A1A=XLAM
      INDE=2
      GO TO 486
D 489 A1A=XLAM-.5
      INDE=3
      GO TO 488
D 499 A1A=XLAM+.5
      INDE=4
      GO TO 486
D 488 A1=A1A+1.
      GO TO 485
D 486 A1=A1A-1.
D 485 A1A=A1
D     AA1=A1A-.5
D     P=0.
D     PPO=0.
D     RESULT=0.
      SENSE LIGHT 0
D402  CLBETA=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)
      IF(CLBETA+80.) 406,406,407
D 406 CBETA=0.
      GO TO 408
 407  IF(A1-1.) 411,412,411
 411  IF(B1-1.) 408,412,408
```

```
D 412 CBETA=EXPF(CLBETA)
  408 IF(X) 52,323,5
    5 IF(X-1.E-14) 323,323,7
D 7   XHH=0.
D     XHH=1.-X
      IF(XHH-1.E-14) 326, 26,8
    8 IF(ABSF(A1-1.)-1.E-8) 433,433,434
  433 IF(ABSF(B1-1.)-1.E-8) 435,435,436
D 435 P=X
      GO TO 87
D 436 C1=B1*LOGF(1.-X)
      IF(C1+45.) 422,422,437
D 437 RESULT=1.-EXPF(C1)
      GO TO 429
  434 IF(ABSF(B1-1.)-1.E-8) 438,438,439
D 438 C1=A1*LOGF(X)
      IF(C1+45.) 423,423,440
D 440 RESULT=EXPF(C1)
      GO TO 429
  439 IF(A1-1000.) 416,416,417
 417  XX=2.*A1*(1.-X)/X
      DF=2.*B1
      PRINT 1995,K12,K
 1995 FORMAT(29H0 CHIX        XIMATION USED IN 219)
      CALL CHI9(XX,DF,P        ,K12,K15)
D     RESULT=1.-PRO
      IF(RESULT-.99999999) 429,420,420
D420  RESULT=1.
      GO TO 429
 416  IF(B1-1000.) 418,419,419
 419  XX=2.*B1*X/(1.-X)
      DF=2.*A1
      PRINT 1994,K12,K15
 1994 FORMAT(29H0 CHIX APPROXIMATION USED IN 219)
      CALL CHI9(XX,DF,PRO,OR,K12,K15)
D     RESULT=PRO
      IF(RESULT-.99999999) 429,420,420
  418 CONTINUE
      IF(A1-1.) 457,457,458
  457 IF(B1-1.) 459,458,458
  458 CONTINUE
      IF(X-.50) 472,473,473
D 473 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      SENSE LIGHT 3
  472 CONTINUE
      IF(A1-1.) 452,453,453
D 452 A1=A1+1.
D     B1=B1-1.
      SENSE LIGHT 1
  453 CONTINUE
   70 II=80
D     FN=A1+B1-1.
```

```
D       XX=A1-1.
D       FMOD=FN*X
D       FO=LOGF(X)/5.
D       FO=EXPF(FO)
D       FMO=FMOD*FO
        IF(XX-FMO+2.) 425,447,447
D 425   XX=FN-XX
D       X=1.-X
  442   SENSE LIGHT 2
  447   CONTINUE
D       AA=X/(1.-X)
D       XXI=II
D       SS=((FN-XXI-XX)*(XX+XXI))/((XX+2.*XXI-1.)*(XX+2.*XXI))
D       SS=SS*AA
  108   II=II-1
D       AI=II
D       D1=(AI*(FN+AI))/((XX+2.*AI+1.)*(XX+2.*AI))
D       DD=D1*AA
D       C1=((FN-AI-XX)*(XX+AI))/((XX+2.*AI-1.)*(XX+2.*AI))
D       CC=C1*AA
D       SS=CC/(1.+DD/(1.-SS))
        IF(II-1) 109,109,108
D 109   SS=1./(1.-SS)
D       C1=ZLOGGM(FN+1.)-ZLOGGM(XX+2.)-ZLOGGM(FN-XX)+(XX+1.)*LOGF(X)+(
       1XX-1.)*LOGF(1.-X)
D       SSUM=LOGF(SS)
D       SSUM1=SSUM+C1
        IF(SSUM1+80.) 423,423,110
D 423   RESULT=0.
        GO TO 421
D 110   SUM=EXPF(SSUM1)
D       RESULT=SUM
        GO TO 421
D 323   RESULT=0.
D       P=RESULT
        GO TO 482
D 326   RESULT=1.
D       P=RESULT
D       ORD=0.
        GO TO 482
  421   IF(SENSE LIGHT 2) 426,428
D 426   X=1.-X
D       XX=FN-XX
D       T1=ZLOGGM(FN+1.)-ZLOGGM(XX+1.)-ZLOGGM(FN-XX+1.)+XX*LOGF(X)+(FN
       1*LOGF(1.-X)
        IF(T1+45.) 460,460,461
D 460   RESULT=1.-RESULT
        GO TO 428
D 461   RESULT=1.-RESULT-EXPF(T1)
  428   IF(SENSE LIGHT1) 454,429
D 454   B1=B1+1.
D       C1=ZLOGGM(A1+B1)-ZLOGGM(A1)-ZLOGGM(B1)+(A1-1.)*LOGF(X)+(B1-1.)
       1F(1.-X)-LOGF(A1+B1-1.)
D       A1=A1-1.
        IF(C1+45.) 429,429,456
```

```
D 456 RESULT=RESULT+EXPF(C1)
      GO TO 429
  459 IF(A1-B1) 466,466,467
D 466 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      SENSE LIGHT 1
  467 IF(X-.85) 468,469,469
D 469 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
      IF(SENSE LIGHT 1) 468,470
  470 SENSE LIGHT 1
  468 CONTINUE
D     Z=1.
D     C2=A1+1.
D230  SUMG=0.
D     SUMG=CLBETA+A1*LOGF(X)-LOGF(A1)
      IF(SUMG+80.) 295,295,240
D 240 RESULT=RESULT+EXPF(SUMG)
D 245 C16=1.-B1
D     SUMH=0.
D     SUMH=CLBETA+LOGF(C16)+C2*LOGF(X)-LOGF(C2)
 250  IF(SUMH+45.) 295,295,260
D 260 RESULT=RESULT+EXPF(SUMH)
D     C17=A1+Z
D     C18=Z+1.-B1
D     C19=A1+Z+1.
D     C20=Z+1.
D     SUMH=SUMH+LOGF(X)+LOGF(C17)+LOGF(C18)-LOGF(C19)-LOGF(C20)
D     Z=Z+1.
      GO TO 250
  295 CONTINUE
      IF(SENSE LIGHT 1) 471,429
D 471 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
D     RESULT=1.-RESULT
  429 CONTINUE
      IF(SENSE LIGHT 3) 474,475
D 474 Y=A1
D     A1=B1
D     B1=Y
D     X=1.-X
D     RESULT=1.-RESULT
  475 CONTINUE
D     RCOM=1.-RESULT
      IF(RCOM-1.E-12) 422,422,427
D 422 RESULT=1.
D 427 P=RESULT
   87 CONTINUE
D     ORD1=CLBETA+(A1-1.)*LOGF(X)+(B1-1.)*LOGF(1.-X)-LOGF(2.)
```

```
      IF(ORD1+45.) 101,101,102
D 101 ORD=0.
      GO TO 482
  102 IF(ORD1-80.) 415,103,103
D 103 ORD=.99999999E30
      GO TO 482
D 415 ORD=EXPF(ORD1)
  482 IF(FLAM-1.E-7) 443,443,444
D 443 PA=P
      GO TO 89
 444  CFLAM=1000.
      IF(FLAM-CFLAM) 483,448,448
 448  PRINT 449,K12,K15
 449  FORMAT(65H0 NORMAL APPROXIMATION WAS USED FOR POISSON PART OF N
     1ENTRAL T 2I9)
      AA=(AA1/FLAM)**.33333333-1.+2./(9.*FLAM)
      BB=AA/SQRTF(2./(9.*FLAM))
      CALL NORM9(BB,CC,PP1)
      PPO=PP1
      GO TO 401
D 483 C1=AA1*LOGF(FLAM)-ZLOGGM(AA1+1.)-FLAM
      IF(C1+45.) 450,450,451
D 450 PPO=0.
      GO TO 401
D 451 PPO=EXPF(C1)
 401  CONTINUE
D     POSUM=POSUM+PPO
D     SUBRES=P*PPO
D     SUBORD=PPO*ORD
D     FIORD=FIORD+SUBORD
D     FIRES=FIRES+SUBRES
      GO TO(494,497,494,496,476,476),INDE
  476 IF(PPO-1.E-12) 484,484,488
  494 IF(PPO-1.E-5) 484,484,488
  497 IF(A1A-1.9) 484,484,498
  498 IF(PPO-1.E-12) 484,484,486
  496 IF(A1A-1.4) 484,484,498
  484 GO TO(487,490,499,491,490,491),INDE
D 490 FRES1=FIRES
D     FORD1=FIORD
D     FPOSM=POSUM
D     CPP=(1.-FPOSM)/2.
D     FIRES=0.
D     FIORD=0.
D     POSUM=0.
      GO TO(487,489,499,491,464,491),INDE
  491 IF(PP-CPP) 492,493,493
D 492 FRES=(1.-FPOSM+FRES1-FIRES)/2.
D     FORD=FIORD-FORD1
      INNE=1
      GO TO 481
D 493 DCPP=ABSF(PP-CPP)
      IF(DCPP-1.E-8) 403,403,404
  403 XQ(KA)=0.
      GO TO 57
```

```
D 404 FRES=(1.-FPOSM+FRES1+FIRES)/2.
D     FORD=FIORD+FORD1
      INNE=2
  481 CONTINUE
D     PA=FRES
D     ORD=FORD
   89 CONTINUE
D     ALPHA=1.
      IF(PP-CPP) 565,566,566
  566 IF(PA-CPP) 567,567,568
D 567 PA=CPP
D     X=0.
      GO TO 568
  565 IF(PA-CPP) 568,567,567
  568 CONTINUE
      IF(I1-1) 547,547,548
 547  INDEX=1
 548  GO TO(532,533,534),INDEX
 532  IF(ORD) 522,522,529
  529 GO TO(569,570),INNE
D 569 XC=X+(PA-PP)/(ALPHA*ORD)
      GO TO 571
D 570 XC=X-(PA-PP)/(ALPHA*ORD)
  571 CONTINUE
D     DEL=0.
D     ABDP=ABSF(PA-PP)
D     ABDP=ABDP/PP
      IF(ABDP-1.E-8) 518,518,520
D518  AX=XC
      GO TO 600
D520  XHH=0.
D     XHH=1.-XC
      IF(XHH-1.E-30) 557,557,556
 556  IF(XC-1.E-30) 558,558,559
D557  X=(X+1.)/2.
      GO TO 560
D558  X=X/2.
      GO TO 560
D559  X=XC
 560  INDEX=1
      IF(I1-N) 516,516,522
 522  NCT=0
      IF(X) 121,121,122
D 121 X=HX
D 122 HX=X
      IF(X-1.E-30) 550,550,551
D551  XHH=0.
D     XHH=1.-X
      IF(XHH-1.E-30) 553,553,552
D550  X=.01
      GO TO 552
D553  X=.99
  552 GO TO(572,573),INNE
  572 IF(PA-PP) 537,536,535
  573 IF(PA-PP) 535,536,537
```

```
D535   DEL=1.-X
       GO TO 538
D537   DEL=X
       GO TO 539
 538   IF(NCT-12) 540,540,546
D540   ABDP=ABSF(PA-PP)
D      ABDP=ABDP/PP
       IF(ABDP-1.E-8) 536,536,541
D541   DEL=DEL/10.
       NCT=NCT+1
       IF(ABSF(DEL)-1.E-14) 536,536,542
D542   X=HX+DEL
D      HX=X
       IF(X-1.) 562,561,561
D561   X=1.
 562   CONTINUE
       INDEX=2
       GO TO 527
  533  GO TO(574,575),INNE
  574  IF(PA-PP) 539,536,542
  575  IF(PA-PP) 542,536,539
 539   IF(NCT-12) 543,543,546
D543   ABDP=ABSF(PA-PP)
D      ABDP=ABDP/PP
       IF(ABDP-1.E-8) 536,536,544
D544   DEL=DEL/10.
       NCT=NCT+1
       IF(ABSF(DEL)-1.E-14) 536,536,545
D545   X=HX-DEL
D      HX=X
       IF(X) 563,563,564
D563   X=0.
 564   CONTINUE
       INDEX=3
       GO TO 527
  534  GO TO(576,577),INNE
  576  IF(PA-PP) 545,536,538
  577  IF(PA-PP) 538,536,545
D536   AX=X
       GO TO 600
D546   AX=X
       PRINT 524,K12,K15
 524   FORMAT(13H0 IN ELEMENT I4,I4,61H FULL CONVERGENCE WAS NOT ATTAI
      1. MAY BE SOMEWHAT UNPRECISEL
 600   CONTINUE
D      YXD=AX
D      AHH=1.-YXD
       IF(AHH-1.E-30) 14,14,13
   14  IF(PP-CPP) 20,21,21
   20  XQ(KA)=-.99999999E30
       GO TO 57
   21  XQ(KA)=.99999999E30
       GO TO 57
D  13  XDD=AAD*(YXD/(1.-YXD))
D      XDD=SQRTF(XDD)
```

```
      IF(PP-CPP) 11,22,22
   11 XQ(KA)=-XDD
      GO TO 57
   22 XQ(KA)=XDD
      GO TO 57
 52   PRINT 51,K12,K15
  51  FORMAT(26H0 ARGUMENT NOT ADMISSIBLE I4,I4)
   53 XQ(KA)=-0.
  57  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CHI9(X2,G,P,ORD,K12,K15)
      DIMENSION G(1),X2(1),P(1),ORD(1)
      KA=1
D     RESULT=0.
D     X=0.
D     G2=0.
      G2=G(KA)
      X=X2(KA)
      IF(G2) 199,199,100
 100  IF(X) 199,90,90
  90  IF(X-1.E-8) 171,171,101
 171  P(KA)=0.
      IF(G2-2.) 164,165,166
 164  ORD(KA)=.99999999E30
      GO TO 200
 165  ORD(KA)=1.
      GO TO 200
 166  ORD(KA)=0.
      GO TO 200
 101  IF(G2-1000.) 168,170,170
 168  IF(X-2000.) 167,169,169
 169  P(KA)=1.
      ORD(KA)=0.
      GO TO 200
 170  Y1=LOGF(X/G2)/3.
      Y1=EXPF(Y1)
      Y2=1.-2./(9.*G2)
      Y3=SQRTF(2./(9.*G2))
      XX=(Y1-Y2)/Y3
      CALL NORM9(XX,P(KA),ORD(KA))
      GO TO 200
 167  IF(G2-4.) 135,135,136
 135  G11=G2/2.+5.E-8
      K=XINTF(G11)
D     THETA=0.
      THETA=G2/2.-FLOATF(K)
      IF(THETA-1.E-7) 145,145,146
 145  THETA=0.
 146  CONTINUE
D     A=THETA*LOGF(X)-X/2.-(1.+THETA)*LOGF(2.)-ZLOGGM(1.+THETA)
D     A3=A
      IF(A+80.) 103,103,102
D102  A2=EXPF(A)
D     T3=A2
D     ORD2=A2
D     T2=0.
      IF(THETA) 130,130,131
D130  A3=A3-LOGF(X)
      GO TO 132
D131  A3=A3+LOGF(2.)+LOGF(THETA)-LOGF(X)
 132  IF(A3+80.) 109,109,108
 108  IF(A3-80.) 162,162,163
 163  ORD1=.99999999E30
      GO TO 104
D162  A2=EXPF(A3)
```

```
D       ORD1=A2
        GO TO 104
D109    ORD1=0.
        GO TO 104
D103    T3=0.
D       ORD2=0.
D       ORD1=0.
D       T2=0.
 104    I=1
 105    I=I+1
D       XI=I
D       A=(XI+THETA)*LOGF(X/2.)-LOGF(X)-X/2.-ZLOGGM(XI+THETA)
        IF(A+80.) 107,107,106
D106    A2=EXPF(A)
D       ORD3=A2
D       T2=T2+A2
 107    IF(I-K) 110,111,111
 110    GO TO 105
 111    IF(THETA) 138,138,139
 139    IF(X-5.) 148,148,149
 148    I=1
D       T11=1.
        N=0
D       A=LOGF(X/2.)+LOGF((1.+THETA)/(2.+THETA))
        IF(A+80.) 113,113,112
D112    T11=T11-EXPF(A)
 113    J=-1
 114    I=I+1
D       XI=I
        J=-1*J
D       A3=(1.+THETA)/(XI+THETA+1.)
        IF(A3-1.E-8) 143,143,147
D147    A=XI*LOGF(X/2.)-ZLOGGM(XI+1.)+LOGF(A3)
        IF(A+80.) 143,143,144
D143    T12=0.
        GO TO 119
D144    T13=EXPF(A)
        C=FLOATF(J)
        T12=SIGNF(T13,C)
D       T11=T11+T12
        N=N+1
 115    IF(N-50) 116,117,117
 116    GO TO 114
 117    PRINT 118,K15,K12
 118    FORMAT(13HO IN ELEMENT I4,I4,61H FULL CONVERGENCE WAS NOT ATTAINED
       1. MAY BE SOMEWHAT UNPRECISE)
 119    IF(T11) 133,133,134
 133    GO TO 121
D134    A=(1.+THETA)*LOGF(X/2.)+LOGF(T11)-ZLOGGM(2.+THETA)
        IF(A+80.) 121,121,120
D120    T1=EXPF(A)
        GO TO 122
D121    T1=0.
        GO TO 122
D138    A=-X/2.
```

```
[illegible]
      A2=0.
      I=0
 153  I=I+1
D     XI=I
D     A=-(13.*X)/XI+(THETA+1.)*LOGF(13.*X/XI)-ZLOGGM(1.+THETA)-LOGF(XI)
      IF(A+80.) 150,150,151
D150  T11=0.
      GO TO 152
D151  T11=EXPF(A)
D152  A2=A2+T11
      IF(I-N) 153,154,154
D154  B=1.01282051+THETA/156.-X/312.
      IF(B) 158,155,160
D158  B=ABSF(B)
D     A=-X/2.+(THETA+1.)*LOGF(X/2.)+LOGF(B)-LOGF(52.)-ZLOGGM(THETA+1.)
      IF(A+80.) 155,155,161
D161  C=-EXPF(A)
      GO TO 157
D160  A=-X/2.+(THETA+1.)*LOGF(X/2.)+LOGF(B)-LOGF(52.)-ZLOGGM(THETA+1.)
      IF(A+80.) 155,155,156
D155  C=0.
      GO TO 157
D156  C=EXPF(A)
D157  D=A2+C
D     T1=1.-D
 122  IF(G2-2.) 123,124,124
D123  RESULT=2.*T3+T1
      ORD(KA)=ORD1
      GO TO 196
 124  IF(G2-4.) 125,126,126
D125  RESULT=T1
      ORD(KA)=ORD2
      GO TO 196
D126  RESULT=T1-2.*T2
      ORD(KA)=ORD3
 196  IF(RESULT) 194,194,195
 194  P(KA)=0.
      GO TO 200
 195  P(KA)=RESULT
      GO TO 200
D 136 X=X/2.
D     G2=G2/2.
D     ORDL=-X+(G2-1.)*LOGF(X)-ZLOGGM(G2)
      IF(ORDL+60.) 172,172,173
  172 ORD(KA)=0.
      IF(X-G2+1.) 174,175,175
  174 P(KA)=0.
      GO TO 200
```

```
      [illegible]
      [illegible]
      [illegible] 180,180,179
D 180 [illegible]
D     [illegible]
      GO [illegible] 135
  179 II=50
D     XI=II
D     SS=X+(XI-Q2)/XI
  176 II=II-1
D     XI=II
D     SS=X+(XI-Q2)/(1.+XI/SS)
      IF(II-1) 177,177,176
D 177 SS=X/SS
D     PROB=ORDA*SS
D     PROB=1.-PROB
      P(KA)=PROB
      GO TO 200
  199 PRINT 198,K15,K12
  198 FORMAT(20H0 ARGUMENT NEGATIVE I4,I4)
      P(KA)=-0.
      ORD(KA)=-0.
  200 CONTINUE
      RETURN
      END
```

```
       [illegible] GAMMA [illegible]
       [illegible] ROUTINE [illegible] STATEMENT HAVING [illegible]
       [illegible] ALSO DOUBLE PRECISION,IN AC AND MQ,AND IN [illegible]
       [illegible] AC FOR DOUBLE PRECISION
       [illegible]
       X=XX
D      Y=X
       IF(X) 900,900,98
  900  PRINT 90
   90  FORMAT(1H0,33X,55HNEGATIVE OR ZERO ARGUMENT ENCOUNTERED IN LOGGM
      1ROUTINE)
       GO TO 201
D 98   TERM=1.0
  99   IF(X-18.) 100,100,101
D100   TERM=TERM*X
D      X=X+1.
       GO TO 99
D101   ZLOGGM = (X-0.5)*LOGF(X)-X+1./(12.*X)-1./(360.*X**3)+1./(1260.*X
      1**5) - 1./(1680.*X**7)+0.5*LOGF(6.2831853071795864)-LOGF(TERM)
D      X=Y
D200   ZLOGGM=ZLOGGM
 201   RETURN
       END
```

```
      . .  ..  *EXPF(-(X(IJ)*X(IJ)/2.))
      Z(I      ) /2.50662712
      XA    ABS (X (IJ))
      IF (XA - 2.5) 105,106,106
 106  U = 1./(XA+1./(XA+2./(XA+3./(XA+4./(XA+5./(XA+6./(XA+7./(XA+8./(XA
     1+9./(XA+ . ./(XA+11./(XA+12./XA))))))))))))
      IF(X(IJ)) 107,108,108
 107  P(IJ) = U*Z(IJ)
      GO TO 101
 108  P(I ) = 1. - U*Z(IJ)
      GO TO 101
 105  ET = 1.41421356/(1.41421356+0.3275911*ABSF(X(IJ)))
      U = C * ((((0.94064607*ET-1.287822451*ET+1.259695131*ET-0.25212866
     18)*ET+0.225836846)*ET
      IF(X(IJ))  102,103,103
 102  P(IJ) = U/2.
      GO TO 101
 103  P(IJ) = 1. - U/2.
 101  RETURN
      END
```

APPENDIX II

Towards Design and Evaluation of Indexing Systems for Information Retrieval

Part I: Costs and Parameters

by

P. Reisner

TABLE OF CONTENTS

I. Introduction

Under contract AF 30(602)-3303, an adaptive thesaurus system for information retrieval is being developed (1). The basic idea of the system is to:

1. Index documents by a simple auto-indexing procedure.
2. Present to a querist, using a man-machine system, a list of "synonymous" cross-references to use in formulating a retrieval question.
3. Use the cumulated experience of the querists to help create these lists of cross-references.

This system is intended as one possible approach to the problem of indexing and retrieving documents. There are, of course, many other approaches. At the present state of our knowledge, however, criteria for choosing between them are still unformulated.

Not only are criteria for choosing a solution to the problem unformulated, the nature of the problem itself is in may cases not clearly understood. This paper, therefore, first attempts to describe the indexing problem in some detail. It then discusses, broadly, possible kinds of solutions to the problem. Considerable research is needed before it will be possible for an information retrieval system designer to choose between these various solutions. The sequel to this paper (Costs and Parameters) discusses the system designer's decisions in greater detail and outlines some experiments needed before he can be helped to specify a suitable indexing system for a given application. The adaptive thesaurus system being developed under this contract is a potential tool for some of this needed experimentation.

II. General Description of the Information Retrieval Process

The information retrieval process, in general, is a communication process which works as follows: there is a set, D, of documents or of items of information of some kind. These documents are to be labelled (indexed) by an indexer or librarian (or possibly by a machine) by selection of one or more symbols or terms from an indexing code, C. These terms are usually some subset of English--e.g., the subject headings of a card catalog. The code is not intended to represent all the "information" in the document, but to serve as a reduced representation of it--a tag, or name. These tags are then stored for an indefinite period of time (as in a card catalog, or in a computer). At some future time, a querist (e.g., library user) interrogates the system, formulating a search question by selection of a term or terms* from

*For details of techniques for combining words in a query (query "grammar"), see (2).

this same indexing vocabulary. (e.g., I want information on [illegible] cats.)
The terms in the search question are then matched against the terms in the storage device. Documents--or information about them (title, author, etc.)--tagged by these terms are then obtained and examined by the querist. The task of the indexer in this process is to label a document so that it can be found by a querist without excessive work.

III. What is the Indexing Problem?

Why is this indexing process a problem?* The task of the indexer seems straightforward enough: find out what a document is about and select, from the indexing code, C, the correct label or labels to describe the contents of the document. Viewed in this way, there are two steps to the process: (1) an identification step, in which the contents of the document are determined, and (2) a labelling step, in which a label is applied to the contents.

Unfortunately, although this is a common way of looking at the process, the above description is very misleading. The "content" of a document should not, as the description assumes, be considered as an entity (or even as a collection of entities). The "contents" of a document are not "things" contained in the document.

*See Appendix I for a discussion of "what is not the indexing problem".

Content is no more inherent in a document than meaning is inherent in a word*. The "identification" step in the identify-and-label indexing process cannot then be considered as a simple process of finding something in a document which is there independently of the person looking and the process used to look.

Rather than identification, then, the task of the indexer is prediction and selection. Instead of "what is in this document," he must answer the question, "what might this document be wanted for?". One of the basic questions for research on indexing for information retrieval is: on what criteria should such prediction be based? **

Determination of document content, then, must be user-oriented, based on the predicted use of a document. *** Once content has been determined, however, there will still be a problem--choosing the correct label for the contents identified.

Labelling is a problem because there is no natural one-to-one correspondence between contents and terms. One term can

---

*The analogy "content is to document as meaning is to word" is quite suggestive and points up the oversimplification in the "identify the contents of the documents" dictum.

**A possible approach to determining these criteria is suggested in the sequel to this paper (2).

***If such prediction proves impossible, then the simplest possible indexing procedure should be used.

have several meanings and several terms can have the same, or nearly the same, meaning. This fact of language can cause failure to retrieve desired documents and retrieval of non-desired ones. So "choosing the correct label" must involve not only an algorithm for choosing, but also specification of a set of labels from which to choose. A second task for research in indexing is the specification of such a set of labels.*

In the next paragraphs, these assertions (1) content is not inherent in a document and (2) there is no one-to-one correspondence between content and labels, are elaborated. The significance for information retrieval of these problems is then indicated.

Content Analysis

It seems intuitively obvious that the content of a document is not an absolute, and that for practical purposes, what-a-document-is-about depends in part on the person reading it. There is, however, some indirect experimental evidence to confirm this. Experiments have been performed to determine inter-indexer consistency in the assignment of index terms to documents. Essentially, in these experiments, the same set of documents is presented to different indexers with instructions to index them. Index

*Section IV of this paper describes some labelling systems and the sequel to this paper discusses the costs involved in each.

terms selected by different indexers for the same document are then compared. These experiments show that there is considerable lack of agreement between indexers in the terms chosen (furthermore, the terms chosen are not necessarily synonyms--which would indicate variability in the labelling process only, not in the selection-of-contents process).

What is in a document, then, depends to some extent on the reader. The task of the indexer, however, is not merely selection, from a finite set of independent well-defined "topics", those that a reader will desire. (The word "topic" is used here as a short notation for element-of-content. It is no more precise than the notion of content itself). A document is not the sum total of a set of discrete, identifiable components. It is not constructed of separable, independent units as a house is constructed of bricks. (It is constructed of words, true, but the relation between word and "topic" is far from clear.)

To make this concrete, let us look at a few examples. Take a document entitled, for example, "Feeding Habits of Cats in Outer Mongolia". This document, for different users, could be about such "topics" as "Feeding Habits of Mammals", or "The Flora and Fauna of Outer Mongolia", or "Cats", or "Ecology", or "Asiatic

Cats" or "Predatory Felines", or "Pets", or "Experimental Techniques used to gather Data on Feeding Habits", or . . ., etc.

And just as one document can be "about" many topics (whose relation to each other is complex), one "topic" can be contained in many documents. If a user wants information on "Feeding Habits of Animals", this information might be found in documents on "Feeding Habits of Cats", or "Cats of Outer Mongolia", or "Dog Diets", or . . ., etc.

To summarize this section, then, documents are not divided into distinct and identifiable "topics", or units-of-contents. For practical purposes, what is in a document depends (partly) on who is to read it. The problem of identifying document "contents" is thus somewhat analogous to identifying word "meaning". Luckily, in most information retrieval systems, there is a "context" of potential users and uses to circumscribe and orient such identification.

Labelling

Exactly analogous problems occur with the words, or terms that are used to label topics as occurred with the topics themselves. Given that one has a topic on which information is desired, then this topic can be described, or expressed, by several terms or combinations of terms. And, on the other hand, one term can be used to represent many topics. In other words, just as content is not inherent in a document, meaning* is not inherent in a term. Furthermore, just as there is no one-to-one correspondence between topic and document, there is no one-to-one correspondence between topic and term.

It is well-known that language permits a variety of descriptions for the same** thing or idea. As an example of this, a table could be described as "a thing with four legs and a horizontal board across them", or "a piece of furniture used for eating", or "a flat object supported by vertical columns", etc. And just as one item can be described in many ways, one description or label can stand for many different objects and kinds of objects--eating table, table of physical contents, steel table, wooden table, etc.

Practical Significance of the Content Analysis and the Labelling Problems

The practical significance of this quite well-known state of affairs for information retrieval is fairly obvious. If language provides alternative ways of describing an object (document), and, if an indexing and retrieval procedure is based on the necessity of

*By "meaning", we mean the relationship between topic and term.

**The notion of "same", too, is somewhat fuzzy. When is one idea "the same" as another?

an exact match between the descriptions of the object by two different people (indexer and user), then we are clearly in trouble. And if, furthermore, a term or a statement in a language can be used to describe many different objects, the trouble is compounded. (We are clearly impinging here on questions which the relevant disciplines have not yet answered--how is language learned and used--by individuals and by groups--what is a concept, how are concepts formed, what are the functions of ambiguity--both syntactic and semantic--and of redundancy in language, etc.) The problem is compounded still further in the case of describing documents in an information retrieval system. Here we are not just trying to describe a thing, uniquely and unambiguously, (as a table) but we are trying to describe it to a person who has never seen it and who does not know whether it exists.

Summary

"Content" then is not inherent in a document. Labels for content do not naturally stand in one-to-one relation to contents.

Indexing, therefore, is a problem because the indexer (and/or the indexing system designer) must predict, not identify:

1. what a document will be wanted for (the user-defined contents), and

2. how this content will be linguistically described by future requestors.

Very little work has been done on the first problem. The field of information retrieval abounds with simple and complex algorithms for "identification of document contents" (select all high frequency words, select everything but high and low frequency words, use relative frequency, etc.) but no valid attempt has been made to test these algorithms or even to state their underlying assumptions. A possible approach to exploring this problem is suggested in (2).

The second problem, the labelling of contents, is discussed broadly in the following section and in greater detail in (2).

IV. On Indexing Languages

To index documents, there must be:

1. A procedure for predicting the "contents" of a document.
2. A "language" in which to label such contents and to phrase queries.
3. A procedure for using the language in the labelling and querying processes.
4. A search process, in which document labels are matched to query labels.

All indexing "languages"* contain the first, and many contain the rest of the following components:

a. A vocabulary, or set of labels.

b. An indexing "grammar" or set of rules for combining the labels into larger labels for indexing.

c. A query "grammar", or set of rules for combining the labels into larger query phrases.

d. A method of semantic control, which may or may not be embodied in the vocabulary.

In addition, there must be two "translation" processes, one for transforming the document-content specification into the indexing language, and another to transform the query into the indexing language.

There are many forms of indexing languages, ranging from those whose vocabulary is limited and precisely defined to those whose vocabulary is that of natural language, and from those with almost no "grammar" to those with fairly complex specialized

*Calling an indexing language a language may raise some protests. The term is used here because: (1) It is common in the jargon of IR, and (2) the relevance of linguistics to information retrieval is vaguely sensed by the author and others but is still not explicit. By the accident of juxtaposition, this relevance may become clearer.

grammars built for specific scientific fields. Criteria for choosing between these various languages are as yet unformulated. In this section, we discuss, broadly, the various kinds of indexing "languages".

Vocabulary

The vocabulary is, of course, simply all the acceptable index terms. There are many designations for such a set of terms (key-words, subject headings, index terms, descriptors*, etc.). The vocabulary may be determined a priori and listed explicitly (as in conventional subject headings, and descriptor systems) or it may consist, potentially, of all English words. Sometimes there may be a list of excluded words (concordances, full text scanning, auto-indexing based on frequency-counting, KWIC, etc. using this method of vocabulary determination) (4), (5), (6). The vocabulary may bear little relation to English terms (Western Reserve's semantic codes are an example of such an artificially constructed vocabulary).

There are variations, too, in the length of the vocabulary "unit". In some systems, a "word" in the indexing vocabulary will

*The term "descriptor" was created by Calvin Mooers (3) to indicate an indexing term plus a definition of the term. In common usage, however, the term is often used as simply index term.

be exactly the same as an English word, determined by the usual criterion of a-string-of-letters-set-off-by blanks. Other systems, recognizing that the "word" in English does not necessarily correspond to our intuitive feeling about the word "units" of language, use larger English phrases or word combinations, such as "mechanical translation", as a unit in the indexing language (P. Baxendale suggests adjective-noun combinations as the "units" and has a program which detects them). But whatever the unit size and whatever the criteria for inclusion or exclusion of a term in the indexing system, there must, quite obviously, be a vocabulary of terms to choose from. Criteria for constructing such a vocabulary, however, are not yet defined and construction of present systems is necessarily ad hoc. Specifying such criteria is one of the most important research problems for information retrieval research.

Grammar

Indexing languages have, not only a vocabulary, but a grammar, or method of combining terms, as well. In printed book-type indexes, the grammar is very primitive, indicating only that two terms are related and what the direction of the relationship is. The "syntactic" devices used to indicate this information are physical proximity and indentation--proximity indicating that two terms are related and indentation that the indented term is subordinate to the main term or limits it in some way. Thus if we have:

abstracting

by author

consistency of

the indented terms act as limiting adjectives modifying the term "abstracting".

In some of the non-conventional indexing systems that have now become quite standard, syntactic devices are used that are in some sense weaker than the above. In the so-called coordinate indexing systems, documents are indexed by single terms. In queries, terms are related to each other by logical "ands", "ors", and "nots"* (e.g., everything on mechanical and translation or on machine and translation but not on chemistry). This grammar permits one to indicate merely that terms are related but not the direction or nature of the relationship.

*To maintain the "language" analogy, only the "and" operation should be considered a syntactic device (it indicates phrasing--that two terms belong together). The "or" operation is related to the "semantics" of the language rather than to its grammar. (It indicates permissible semantic substitutions). The "not" operation defines a context (by elimination).

Other grammatical devices can and have been used. Sometimes there is an explicit list of the kinds of grammatical relations* that can hold between two terms - e.g., process-thing processed, or input-output, agent-object acted on, etc. These relations can be indicated either by defining, within the language, a separate class of relational words (like verbs), or by defining a class of affixes to be directly attached to the index terms. Some of the coordinate indexing systems use such affixes or "roles" (7). These are analogous to derivational endings in natural languages.

In natural language, we have grammatical devices for indicated sentence-hood and/or phrase-hood--that certain terms "belong" together. Some coordinate indexing systems also have such a device to indicate phrasing - called a "link." Links are essentially affixes attached to each of the words that belong together (7). Thus, phrasing is also indicated in the "morphology" of the language.

In systems which use full text scanning, grouping or phrasing exists naturally in the index (which is here the full text), and queries can then be formulated using some of this information, e.g., find all documents in which word "a" occurs in the same sentence as word "b."

---

* This kind of relation has been called by P. H. Smith "the grammar of the subject area" to distinguish it from the grammar of a sentence (subject-object relationship).

Or relative position can be used - e.g., find all documents in which word "a" immediately precedes word "b." (4)

Our indexing languages then, do incorporate some grammatical devices - phrase indicators, indicators of the direction of a relation between terms, indicators of the kind of relation between terms. One of the problems for I R research is to determine the effect that incorporation of successively "stronger" grammar will have on the effectiveness of the retrieval system.*

Semantics

Our indexing languages then, have a vocabulary and some grammatical devices. They also have devices for handling the "semantic" problem we discussed before (one term can have several meanings and one meaning can be related to several terms). One way to do so is to control, or standardize, the indexing vocabulary. Such control can be indicated, for example, by listing a limited number of acceptable index terms. It can be further indicated by defining the scope of the acceptable index terms. Such definition is somewhat akin to a translation --English-to-indexing-language, i.e., whenever "feline"

*By the strength of a grammar, we mean merely the amount of syntactic information it permits. Thus, a grammar in which one can indicate only that two terms are related is weaker than one which indicates the direction of the relationship, is weaker than one which specifies the kind of relationship, etc.

occurs in the title, use "cat". Another, somewhat looser technique for handling the semantic problem is to build a thesaurus, or explicit list of permissible semantic substitutions in the language (8), (9). Indexer and/or user are then free to choose from these lists. A thesaurus is an outgrowth of the "see also" cross references in conventional library systems. (The "see" references were translations from English to index term). The two methods of handling the semantic problem (controlled vocabulary versus thesaurus) can both be combined in the same system, of course. There is not yet a methodology (or a set of rules) either for standardizing a vocabulary or for constructing a thesauri. There is not even a precise formulation of the desirable characteristics of the end product.

This is, then, another problem for IR research.

V. Relation of Indexing to Machine Translation

We have been talking of indexing "languages". It is then quite natural to talk of translation between natural language and indexing language. And from this, in turn, to suspect a relationship between machine translation and information retrieval. There are very significant differences, however. In machine translation both the "source" and the "target" languages are known and the task is to devise a procedure for translating between them. In information retrieval, we have a two fold task:

1) specifying and designing the target language, and

2) translating

Of the two, the specification of the indexing language is by far the more important problem. We do not yet have sufficient insight to do so intelligently.

Another difference is that in M. T. there is a single translation step whereas in information retrieval there are two: document-language-to-index-language and querists-language-to-index-language. And these two, if retrieval is to be effective, must yield identical results.

It is, of course, obvious that indexing also involves condensation of information (concept formation, in a sense) and MT does not.

There will be some of the same problems (ambiguity for example) since both do deal with language.

For very much the same reasons, too close an analogy between the communication theory model and the indexing and retrieval process could be misleading. There are the following significant differences in the two processes:

1) In the Shannon process, the encoding and decoding apparatuses both function according to the same rules - there is a one-to-one, reversible transformation, i.e., a message or letter, will be encoded at the transmitter into a given sequency of bits. This sequency of bits,

when decoded at the receiver, will (without the effect of noise) yield the original letter. This does not necessarily occur in our Information Retrieval process. There, the encoder, to index a document, will select each symbol from a subgroup* of symbols within the larger set of symbols (i.e., if the document is on "cats," he might encode it with any of the semantic substitutes of cats: "mammals," "tabbycats," "felines," etc.). T e decoder will then select from this same subset of symbols but may not make the same choice. Fundamentally, instead of a coding and decoding process, we have two encoding processes which, while similar, may not be identical.

2) In the Shannon process, the source of error (noise) is either in the physical characteristics of the transmission channel or in the encoding process. In the IR situation, however, the source of error is the code itself (the redundancy and ambiguity of the language).**

VI Motivation for an Adaptive Thesaurus System

The motivation for an adaptive thesaurus system as a possible solution to the indexing problem has been explained in earlier papers (impossibility of determining, a priori, all interrelated "synonym"

---

* The subgroups are not necessarily discrete and distinguishable.

** There may, however, be analogous concepts although the models are different. We need, for example, a notion of the redundancy of a language, defined in terms of the number of terms and the number of permissible semantic substitutions for each term in the language.

pairs, avoidance of reindexing when user vocabulary changes, potential for adaptive reorganization to improve system performance, etc.).

More important than these, however is the potential of a device of this nature as a learning tool. Hopefully, this paper has indicated (in very broad outline) the nature of what we want to learn about indexing. The adaptive thesaurus can be a tool to obtain some of the desired information.

VII Concluding Remarks

The field of Information Retrieval is still trying to define itself. Problems are often ill-formulated and ill-chosen. Complex solutions to problems are often suggested - and worse, implemented - before the nature of the problem is understood. In the area of indexing, which is central to IR, considerable work is being done to devise specific systems for given applications (real or experimental). (In issue 11 of Current R&D in Scientific Documentation (10), one out of every 9 projects in IR claimed to be working on thesauri, alone) - but little of it attempts a clear formulation of the problems or generalization of the potential kinds of solutions. The intent of this work is, therefore, to indicate (to the extent the author understands it) 1) what the indexing problem is 2) what we need to know to choose between various possible solutions 3) how to obtain the desired data.

Appendix: What is not the indexing problem?

Considerable effort is being spent on indexing, particularly on auto-indexing, which seems inappropriate to the problem and disproportionate to the results which can be obtained. Much of it is obscure, the proposed solutions are ad hoc and without either intuitive appeal or rational basis (to the author, at any rate), the assumptions are unstated. The problems discussed below may very well be problems, but they are not central problems, and proposed solutions are often inconsistent with the problems as stated.

1. The time lag, or volume of information problem.

(People can't keep up with indexing and there is a consequent delay in getting documents indexed.) Proponents of auto-indexing justified on this basis, often claim "what people are doing is good enough. We just want to do it faster." Without questioning the criteria for "good enough" (usually unstated), let us look at what is being done to meet this problem. The first thing we notice is that auto-indexing is not as a rule trying to duplicate conventional techniques and results. Such duplication would involve, for example, taking a pre-existing catalog or indexing system (e.g., Dewey Decimal) and looking for terms, in documents, which would uniquely indicate that a document belongs in a given category. (Some work is in progress along these lines but much of the effort in auto-indexing is based on variants of frequency counting procedures.) The second thing we noticed is that much of the work that human indexers do now is clearly mechanizable, very simply, and that most of the methods advocated by auto-indexers seem much too strong for the desired goal. Experiments have shown, for example, that a large percentage (c. 60%) of index terms are contained in titles of documents (1), another 20%-25% are near synonyms. If this is true, then a simple concordance program, augmented with a small thesaurus, or perhaps a program like P. Baxendale's adjective-noun extraction routine, would suffice. Auto-indexing based on complicated statistical manipulations of full text is too strong a tool to use (even if it works). If simply speeding up indexing is the goal, rather than improvement of indexing, then existing simple techniques should be used. If this is the only goal, effort should be expended on input preparation, search problems, etc., rather than on indexing problems.

2. Inconsistency of human indexers

This reason is often given as an advantage of auto-indexing and is certainly part of the problem. However, this inconsistency of human indexers is a symptom of the disease, not the disease itself. Indexers disagree because what is in a document depends on who is reading it. Assuring agreement between indexers would not necessarily assure agreement between indexer and user. Indexer consistency has been achieved before in very simple ways. For example, in "the Index of Christian Art", we read :

> In order to maintain the rigid standards of uniformity and consistency, . . . the index staff has been limited to a few.

There is no guarantee, however, that consistency of indexers achieved in this way, or achieved by using a machine to index documents, will have any effect whatsoever on indexer-user inconsistency--which is really our problem. (e.g., a machine might index a document quite consistently, by the word "fission" whenever this word occurred in the document. However, the user who wanted documents on "nuclear energy" or "atomic energy" would not find it).

3. Indexing depth

By "indexing depth", we mean the number of different terms (corresponding to different "topics") assigned to a document. One of the arguments for auto-indexing usually states: "indexers don't have the time to read a document completely and index it throughly. A machine could read (!) the whole text and index as deeply desired". Unfortunately, however, the "deepest" indexing possible, (full text searching) has already been tried, experimentally, with far from extraordinary results (5). Auto-indexing schemes are usually selection systems--they select, from all the words in a text, some subset to serve as index terms. It seems obvious that if all the words in a text will only give a certain level of retrieval, a smaller number of terms selected from the text will not improve matters. Selection criteria may be needed because of storage limitations. If selection is necessary, however, then rational formulation of selection criteria is required. This is yet to be done.

REFERENCES

1. Reisner, P., "Constructing an 'Adaptive' Thesaurus by Man-Machine Interaction", Final Report on Applied Research Program Aerospace Intelligence Data System (AIDS), Contract 19(626)-10, May 15, 1964, Appendix II B.

2. Reisner, P., "Towards Design and Evaluation of Indexing Systems for Information Retrieval, Part II: Costs and Parameters", Quarterly Report No. 2, Computer Programming Techniques for Intelligence Analyst Application, Contract AF 30(602)-3303,

3. Mooers, C. N., "The Indexing Language of an Information Retrieval System", Information Retrieval Today, Simonton, W. (ed.), University of Minnesota, 1963.

4. Kehl, W. B., Horty, J. F., et. al., "An Information Retrieval Language for Legal Studies", Communications of the ACM, September 1961, pp. 380-389.

5. Swanson, D. R., "Searching Natural Language Text by Computer", Science, October 21, 1960, pp. 1099-1104.

6. Damerau, F. J., "An Experiment in Automatic Indexing", IBM Research Report RC-894, February 19, 1963.

7. Holm, B. E., "Information Retrieval - A Solution", in Chemical Engineering Thesaurus, American Institute of Chemical Engineers, New York, 1961.

8. ASTIA, Thesaurus of ASTIA Descriptors, Second Edition, December, 1962.

9. Wall, E., "Study of Engineering Terminology and Relationships Among Engineering Terms", Final Report, Engineers Joint Council, August, 1963.

10. Current Research and Development in Scientific Documentation, No. 11, National Science Foundation, November, 1962.

11. Kraft, D. H., "A Comparison of Keyword-in-Context (KWIC) Indexing of Titles with a Subject Heading Classification System", American Documentation, Vol. 15, No. 1, January, 1964, pp. 48-52.

Towards Design and Evaluation of Indexing Systems for Information Retrieval

Part II: Costs and Parameters

by

P. Reisner

TABLE OF CONTENTS

I Introduction

Information Retrieval is being viewed increasingly as a system design activity. It is not, however, as generally accepted that the development of the indexing languages* for these IR systems must also be a systems design activity, incorporating the same attitudes, approaches and procedures used for the development and selection of hardware and of programs.

In this paper, therefore, some of the costs related to indexing and some of the "parameters" of indexing languages are isolated. Qualitatively, the language parameters are then related to the costs. This qualitative discussion should be followed by precise formulation and by experimentation to replace the qualitative information with at least, gross indications of the quantitative data involved.

Some of this needed experimentation is then outlined. (One of the experiments has recently been initiated). An adaptive thesaurus system, discussed previously,(1) may serve as a tool to obtain some of the desired data.

This paper is in no sense definitive. It is intended merely to indicate: what we need to learn, why we want to learn it and to some extent, how we might begin to do so.

* For a general discussion of the indexing problem and of indexing languages, see (2).

II Indexing System Costs

System "costs" related to indexing can, for convenience, be divided into three major subcosts:

1. Retrieval costs, C_r
2. Operating costs, C_o
3. Design costs, C_d

The retrieval cost, C_r, would be the "cost" of poor retrieval, measured as a function of "miss" and "trash" as explained below*.

To evaluate an indexing system, or to compare different systems, we would define a total cost as some weighted function of these subcosts. (e.g., $C_{total} = aC_r + bC_o + cC_d$).

Retrieval Costs

Tests and evaluations of indexing systems (3)(4) usually consist of:

1. A set of documents.
2. A set of questions.
3. A procedure for finding, for each question, a subset of the document collection which is "relevant"** to the query.

* Clearly a more positive view could be taken: measure retrieval effectiveness, i.e., how well rather than how badly the system functions. The choice, however, is immaterial (except from a psychological standpoint) since one can easily be transformed into the other.

** The question of what constitutes "relevance" is an unresolved problem. For a discussion of the problem and a suggestion for circumventing it, see Section V, p. 18.

This can be visualized in terms of figure 1, below, where:

A = Total number of documents in the system that would be judged relevant to a query

B = Total number of documents retrieved by a query

C = Total number of documents both retrieved by a query and judged relevant to it. ($C = A \cap B$)

D = Total document collection.

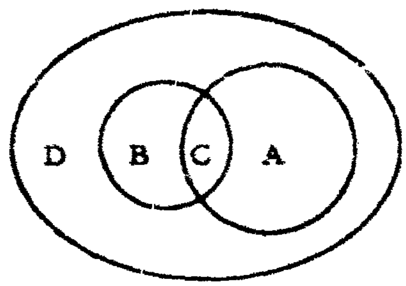

Fig. 2

Various definitions of retrieval effectiveness are employed by different investigators, most of which can be viewed as some function of the above quantities, and transformed into each other for comparison if desired.

Thus Cleverdon (3) and Kochen (5), who use the same functions but with a different terminology, talk of "hit rate" (h), (Cleverdon's "recall") and acceptance rate (a), (Cleverdon's "relevance") where:

$$h = \frac{\text{retrieved and relevant}}{\text{relevant}} = \frac{C}{A}$$

$$a = \frac{\text{retrieved and relevant}}{\text{retrieved}} = \frac{C}{B}$$

Swanson adds a degree of relevance and defines a retrieval score , S , which combines the variables into a single quantity, ($S = R-pI$). Here "R" is "the sum of the relevance weights of the retrieved documents divided by the total sum the relevance weights (for a given question) of all documents in the library; I is the effective amount of irrelevant material (and is given by N-LR where N represents the total number of documents retrieved and L represents the total number of documents in the library); and p is the irrelevance penalty and may take on arbitrarily assigned values"(4). In terms of the diagram, we have, using the square brackets to denote the weighted relevance figures):

$$S = R - pI = R-p(N-LR) = \left[\frac{C}{A}\right] - p\left(B-D\left[\frac{C}{A}\right]\right)$$

Western Reserve defines:

$$\text{effectiveness} = \frac{\text{number of documents found}}{\text{number in collection}} = \frac{B}{D}$$

and

$$\text{precision} = \frac{\text{number of relevant documents}}{\text{total number found}} = \frac{A}{B}$$

Clearly, other functions of these variables are feasible and and it is not our purpose to discuss them here. We prefer* to work

*This preference is based on the ease with which some of our indexing language parameters can be related to these costs.

with the quantities $\underline{T}$ and $\underline{M}$, where:

$$T = B - C$$

and

$$M = A - C$$

T, obviously, is the total number of documents retrieved and not relevant (the "trash") and, M, is the total number of documents relevant but not retrieved (the "miss"). Expressing, for example, Kochen's equations in terms of M and T, we have:

$$h = \frac{A+M}{A} \quad \text{or} \quad \frac{C}{M+C} \quad \text{or} \quad \frac{T+B}{A}$$

$$a = \frac{A+M}{B} \quad \text{or} \quad \frac{C}{T+C} \quad \text{or} \quad \frac{T+B}{B}$$

The "costs" of poor retrieval, then, are "trash" and "miss".

Operating costs (C_o)

These are: data preparation costs (indexing costs) query preparation costs, searching and matching costs, storage costs, etc. and are fairly straightforward. ("Trash," considered here as a retrieval cost, could also be considered as an operating cost. As such, it would be related to the human waste time to eliminate undesirable material. With the arbitrariness of all classifiers, however, we prefer it in the above category.)

Design costs (C_d)

The cost of designing a language system is largely unknown, particularly since the procedure is still unspecified* We will confine ourselves to two values, $C_d = O$ and $C(d) > O$ in the discussion.

III Indexing System Parameters

An indexing system consists of:

1. An indexing language
2. An indexing and retrieval procedure
3. An environment, consisting of a document population and a user population

In the next sections, some of the "parameters" of these subsystems are listed, and in the following sections, they are related to the retrieval, operating and design costs discussed above.

The Indexing Languages.

The components of the language are the vocabulary and the grammar. To these, we add the mechanism for semantic control, which may or may not be implicit in the vocabulary.

Vocabulary

To differentiate one kind of indexing vocabulary from another, we distinguish:

*Preparation of the Engineers Joint Council thesaurus took ~18 months of professional work.

1. The number of terms in the vocabulary.

2. The size of the vocabulary "unit" (word, phrase, etc.).

3. The extent to which vocabulary is standardized or controlled (i.e., does not correspond to natural language vocabulary)

Grammar

A "grammar" for an indexing language, will consist of a set of categories, together with a set of rules for combining terms in these categories for use in an indexing label and in a query phrase. For purposes of comparing indexing languages, we can use the number of grammatical categories in the language. Thus a grammar with only one category will be "weaker" than one with two, etc.

A coordinate indexing system, then, has the weakest kind of grammar because there is only one category. Any term can be combined with any other, using the "and" operation. In these coordinate indexing systems, the only grammatical information incorporated is the information that terms can be related to each other (co-occur in an indexing label and/or in query statement). A slightly stronger "grammar" would be one in which there are two categories, a main-term category and a dependent-term category, for example. Such a grammar would permit the user to indicate, not only that two terms are related, but also the direction of the relationship ("a" modifies "b," or a depends on b, etc.). Still stronger "grammars" would list, explicitly, the nature of the relationships between terms (e.g., processor-thing processed, or input-output, etc.).

Semantics

The semantic problem, as discussed in (2) can be handled by "controlling" vocabulary, by a thesaurus (a set of "synonyms," or semantic substitutes) or by some combination of controlled vocabulary and thesaurus.

Procedures

The procedures are the methods of using the "language" to index and retrieve documents. The distinguishing characteristics of different systems will be both quantitative and methodological--i.e., how many words should be chosen and what procedure should be used for choosing them. The quantitative distinctions apply at two steps in the indexing and retrieval procedure, so we will be interested in:

1. Indexing depth--the number of terms selected from a vocabulary to index a document, and
2. Query depth--the number of terms used to formulate a query.

Some of the methodological distinctions are:

1. The section(s) of the text used for selection of index terms (title, abstract, conclusion, full text)
2. The agent (human or machine)

3. The method used to select terms (syntactic analysis, frequency counts, human judgement)

The Indexing Environment

The environment consists of a document population and a user population. We are concerned only with the linguistic characteristics of these populations*.

Document populations are characterizable by rank-frequency distributions, type-token ratios and other vocabulary dispersion measures. These characteristics are based on the vocabulary of the document texts.

User populations would be characterized by very much the same parameters, but now the "text" would be the questions put to the system by the users.

No work has yet, to the knowledge of the author, been done to characterize this "query language" of the user population. Do different users, for different users, for example, ask for the information in the

*We are interested in characterizing these populations for two reasons: (1) Generalization of results of (future) experiments (i.e., in order to apply results of a test on one population to another population, we need some basis for the judgment that one population is "like" the other). (2) Different user and document populations will probably require different kinds of indexing "languages", and with cost in mind, we will want to know, what kind.?

same way, e.g., does one user ask for "cats" and another for "felines?" In other words, is the user population linguistically homogeneous or heterogeneous?*

The interactions of document language, user language, and indexing "language" should not be forgotten, i.e., relation between index term frequency and query term frequency, number of (user-defined) synonyms per term, number of homographs per term, (in both user and document vocabularies) frequency distributions of synonyms and of homographs, relation between frequency of a term and a number of synonyms it has, etc.

IV. Effect of Indexing parameters on costs

For each decision an indexing system designer will make, he should know, what will it buy and what will it cost? Some of these decisions are qualitatively discussed below. Obviously, such qualitative discussion is meaningless if not followed by precise data.

Language

Decision 1. What should the word units of the language be.

Definition of a word is one of the more persistent linguistic problems. In an artificial indexing "language," the problem has a

* Document and user populations have often been, loosely, described as heterogeneous or homogeneous. These terms are too weak to be useful. At very least, the populations should be defined as homogeneous or heterogeneous with respect to some characteristic. For example, user groups can be homogeneous with respect to: 1) the information desired (John Jones and everything about him) 2) the classes of information desired (man number, salary...) 3) the nature of the output desired (data, ideas, techniques...) 4) the linguistic representation of the information (or classes of information) desired 5) educational level, professional status, etc.

practical significance. The simplest definition, from the point of view of machine identification of word units, is "a word is a string of letters set off by blanks." However, there are larger "units" which function effectively as single terms (e.g., mechanical translation, command and control, operations research). The problem facing an indexing language designer is: should these be considered as units or not?

The "trade-offs" conditioning this decision are: miss, trash, storage space, search time, and data preparation costs. For example, if "mechanical translation" is not considered as a single unit, queries for "mechanical translation" may result in so-called "false drops," or trash* (documents on "translation of mechanical energy" may be retrieved). If "mechanical translation" is considered as a unit, but only as a single unit, the problem of "trash" does not arise, but there is now a possibility of "miss." (The term will be stored in an index, in some order-probably alphabetical-according to one criterion only. Suppose it is stored under M. Then requests for information on "translation," either human or machine, will not retrieve it.)** If "mechanical translation" is considered both as a single unit and as two separate units, the problems of miss and trash are eliminated, but now storage (and possibly search time) are increased. (One document is indexed in three places).

* Other devices can be used to prevent this problem, but all involve "processing work," so the cost of handling this problem is either "trash" or "work."

** There is an alternative to "miss" here, i.e., read the entire file. So our cost is either "miss," or extra "work."

In addition, if the terms are to be considered a "unit" in some system, there are additional "costs" associated with designing the system (deciding what combinations of terms to consider units) and with processing (identifying these units in a text)).

At least gross quantitative estimates are needed to replace this purely qualitative discussion.

Decision 2. How many terms should the indexing vocabulary contain.

Criteria for determining the desirable size of an indexing vocabulary are still not formulated. Assertions and injunctions to document-alists are sometimes made for or against a small vocabulary-but these injunctions are seldom supported-even on an intuitive basis. While the issues are far from clear (to the author) the problem requires formulation and is consequently discussed.

The purpose of designing an indexing language with a small number of terms is the control of the problem of "synonymy." Suppose an indexing language has two terms "a" and "b" which are interchangeable, or "synonymous." Thus an indexer, to label a document, might select "a," while a querist, who might have desired the document if found, might select "b." Thus the user would "miss" the document. In general, however, we are not restricted to only one document and only one pair of terms. For each term in the vocabulary, there could be a set of "synonymous" terms. Thus for a given question, a user would "miss" all documents indexed under each of the synonyms he did not pick, and the amount of his miss would be the sum of the frequencies (number of documents) per non-selected synonym. To avoid this "miss," an index language designer could create a single index term which would replace all of the terms in a set of synonyms by one term. Since there would be only one term to choose, there could then be no problem of "miss" caused by synonyms. If this many-to-one mapping (English terms to controlled index term) were performed for the entire vocabulary there would be a decrease in the total amount of miss for the system. Thus, at least on intuitive grounds, it is clear that the smaller the indexing vocabulary, the smaller the chance of "miss."

However, although miss would decrease, "trash" would increase. The above procedure would be desirable if there were sets of true synonyms - with each term in a set replaceable by all others - for all contexts. This is unfortunately not true. So if, for example, we substitute for English word a and English word b, the index term c, we are decreasing the power of the language to make distinctions. If a given user wants information on a, but not on b, he must invariably receive b as well. For him, all documents indexed under b will be trash.

Thus, as the size of the vocabulary decreases, we can expect miss to decrease but trash to increase.

There are other effects of vocabulary size as well. A decrease in the number of terms in the system would result in a slight saving of of storage space. (If N_f is the number of terms in the original English vocabulary, N_c in the smaller controlled one, and D is the total number of documents indexed in the system, the difference in storage would be the space to store N_f-N_c words. D would require the same amount of storage but would be distributed differently amont the N words.) Human reading time to scan the output and machine read-out time would of course be greater, the smaller the vocabulary, because there would be more documents per term.

Once again, how much?

<u>Decision 3.</u> How should the semantic problem be handled - should it be ignored, should a controlled vocabulary be developed or should a thesaurus be used.

The main issue here is: to what extent do the users agree in the linguistic representation of what they want (i.e., to what extent is the user population linguistically homogeneous). If there is considerable agreement, there is not much of a problem and it can be ignored. ("Considerable" is an arbitrary notion and would need specifying) If there is very little agreement, then nothing much can be done. In the wide range of "some" agreement, the choice is between a controlled vocabulary and a thesaurus.

A controlled, standardized vocabulary can cause "trash," as discussed above. It also requires work - construction of the controlled vocabulary. Costs of data preparation (indexing) are increased (With an uncontrolled vocabulary there is merely a selection process to choose the desired term. With a controlled vocabulary, on the other hand, one must first select an English term, then translate it into the standarized index term).

A thesaurus, however, causes extra work during the search process. (Extra human time to select the additional terms and to formulate an enlarged query, extra machine time to access each individual term, read document numbers into an intermediate store, intersect the lists of documents to eliminate duplication, etc.)

Decision 4: How "strong" a grammar should be incorporated into the language.

The "strength" of an information retrieval grammar was defined in terms of the number of grammatical categories it contained. Thus the simple coordinate indexing system, which did not assign terms to different grammatical categories was the weakest (one category).

Let us take the "roles" sometimes included in coordinate indexing systems as an example of the categories (e.g., input, output, thing processed, catalyst, etc.). If we do not indicate these categories, then it is conceivable that a querist wanting a document on substance a as input might be given a document in which substance a was output. For him, the system would create "trash."

However, if we build a system with roles indicated, the result may be "miss." For example, a document might be indexed as pertaining to substance a as a catalyst. Now a user might want all documents in which substance a was input. A catalyst is, in a sense input. But if the indexer used the "role" catalyst and the querist used "input," the result would be "miss."

It is clear that we are once again faced with a problem of "synonyms" - synonymous categories or roles rather than synonymous terms. As the number of categories increases, the system permits finer distinctions. However, the chance of indexer and user making different choices from the list of categories will increase and so will the chance of "miss." But with a smaller number of categories, there is a greater chance of "trash." So as the number of categories increases miss will miss will increase and trash decrease.

Of course, one could also have a "thesaurus" of substitutable categories. The extra work involved would then be exactly analogous to the extra work involved in use of a thesaurus of index terms.

In addition, inclusion of grammatical devices in the language increases both the costs of indexing and the costs of querying (the roles or categories must be selected as well as the terms).

Procedures

Decision 5. What criteria should be used to index specific documents?

Should terms be selected from the title, the abstract, the body of a text? Should they be selected by syntactic analysis, frequency counts, other complex algorithms? At this stage of our knowledge, these questions are premature. First, it is necessary to specify what we want to select from a document for indexing. Only then can discussing the method of selection be meaningful.

The criterion for the selection should be: what might this document be wanted for by the majority of its users.* To discover, at least on the basis of past usage, how a specific document has been used, might be a sensible starting approach. Use of an citation index for such an exploration is suggested in section V.

V. A Few-Needed Experiments

Indexing Language Parameters

In section III, the effects of several indexing language parameters were qualitatively discussed. Clearly, at least gross indication of the quantities involved should be determined.

Semantics

In this paper, the "semantic" problem associated with indexing was discussed. In a preceding paper, an adaptive thesaurus system was suggested as a possible approach to this problem. To test the adaptive thesaurus idea, and to gather data of the semantic problem, an

*The misleading assertion that one should "identify the contents of a document" to index it was discussed in (2).

experiment has recently been initiated*.

In this experiment, we will present to a user a list of his query terms (his "profile") and ask him to generate synonyms for each term. Then, for each word, we will compile a composite list of all users' synonyms. The composite list is then to be returned to the individual with a request to check those terms he considers synonymous. The terms-plus-synonyms he has checked are then to be inserted into the system and changes in retrieval ("miss") noted.

This experiment should also provide data on the number of synonyms per term, on the extent of linguistic agreement between user, on the relation between number of synonyms for a term and its frequency as an index term, etc.

Relevance

Tests of indexing systems, as a rule, compare a set of documents obtained via the system in response to a query with an ideal set of "relevant" documents. The basic assumption is that there exists some one unique subset of the document collection which is relevant to the query. The documents which are found by the procedure under test are then to be compared to this ideal set of relevant documents. The evaluation is usually expressed in terms of:

1. The number of "relevant" documents obtained.

*The experiment is a collaborative effort between IBM Research and the Advanced Systems Development Division using the SDI (Selective Dissemination of Information) system.

2. The number of "relevant" documents missed (miss).

3. The number of "non-relevant" documents obtained (trash)

However, these tests immediately run into difficulties with the notion of "relevance" on which they are based. The difficulties, as usually states, are that (1) the notion of relevance is vague and there is little agreement between people in the judgment of relevance, and (2) "the relevance of a document to a query admits of degrees". One experiment, at least, has been known to founder because of these difficulties. In 1953, a comparison of the ASTIA system and the Uniterm System of Documentation Incorporated failed because the two groups were unable to agree which documents were relevant (vested interests undoubtedly played a part) (6). Sometimes, while recognizing the existence of the problem, experimenters proceed as if it did not exist. For example, in a report by A. D. Little (7), we read (after a discussion on measurements of retrieval effectiveness):

> "The first two measures have been used previously in the literature on evaluation of retrieval systems. These measures assume that agreement as to what constitutes relevance is possible, but this assumption is of questionable validity. The whole question of the meaning of relevance is, in many ways, obscure. In the definitions given above, it is treated as a black or white matter--a document

either is or is not relevant to a request...Nonetheless, for purposes of the analysis...it is presumed that relevance can be defined as a "yes" or "no" decision from the viewpoint of the user of the retrieved information."

Occasionally attempts are made in experimental investigations to circumvent (rather than explore or solve) these problems. Cleverdon, for example, in his comparative test of indexing systems, derives his question set from documents in the collection and then attempts to retrieve, at least, these so called "source" documents. He then has, for each question, at least one document which, by virtue of the artificial experimental situation, he has defined to be relevant. To handle the problem of degree of relevance for the other documents retrieved, he uses a four point scale (more useful than source document, as useful, of some relevance, no relevance.) In attempting to test an indexing system with real questions, however, the first artifice could not be used. And the four point scale, while it handles the problem of degrees of relevance, complicates the variability-between-people problem still further. For now we have, not only possible variability in a yes-no decision, but variably in the scaled judgments to contend with.

Since this criterion of relevance is central to the evaluation of indexing systems, it merits attention.

The first problem, of course, is to determine the extent of the problem, i. e., 1) to what extent does the judgment of yes-no relevance vary between individuals, and 2) to what extent does the judgment of relevance on a scale vary between individuals. (A third question should be, to what extent is one person, after a time lag, consistent in his judgment.)

Then, if these judgments prove to be so variable that evaluations based on them are unreliable, we can then reformulate our directions to the experimental evaluators to force less subjective responses, as indicated below.

To determine the extent of the problem, we could take a set of questions and the results of these searches on a document collection. "Evaluators" would be needed (perhaps college students would be a better choice than volunteers). A question and its responese could be given to several evaluators, who would be asked to judge relevance or non-relevance.

If the judgment of relevance proves difficult, we could go about this in a different way. Instead of asking for an absolute relevance judgment of each document with respect to the query, independent of the other documents, we could ask the evaluator to take the entire set of responses to a query and rank them. We would thus have a relative judgment (this document is more relevant than that one). We could then compare the ranking.

If we determine that the relevance judgment is, as suspected, a unreliable basis for the evaluation of retrieval systems, we could reformulate our instructions to the examiners to decrease this variability. To do this, we would classify questions into several broad categories, based on the expected type of answer. For each category of question, we would then categorize the potential types of answers. In this way, by asking the evaluator a fairly detailed and concrete set of questions instead of the vague "is the document relevant?" we should force a more precise answer.

By categories of questions, for example, one could use (this is suggestive only, not final),

1) Specific data is desired
2) Compendium of data desired
3) Correlation of data desired
4) Interpretation of data desired
5) Methods, Processes, Procedures desired
6) Survey--what is going on-desired

etc.

Then for each question-type, we could break down the possible answers and ask the evaluators to check, e.g.

For question-type 1 (specific data)

1. Document contains exact and complete data

2. Document contains some of data

3. Document contains information from which data can be derived. etc.

In other words, by giving our users more precise direction, we may get more precise answers.

If the judgment of degree of relevance of a document to a query proves unreliable, we could also force a response here. Instead of asking a user to indicate relevance on a four point scale, we could try to determine how much of the document he has to read before making a judgment of relevance. (e.g., If only the subject heading is read, the document is highly relevant, if the title, it is less so, if the abstract... etc. and so on through full text.).

Since evaluations of indexing depend critically on this relevance judgment, it merits some attention.

Document "Content"

In the jargon of the field of information retrieval, the process of indexing is often seen as "identifying document content," "identifying relevant information," "determining document contents," etc. In (2) this viewpoint was discussed and an attempt made to indicate that: 1) content is not inherent in a document,* and 2) what is in a document, for retrieval purposes, depends on who is reading it.

The task of indexing, then, involves prediction of the probably use of a document. On what criteria should this prediction be based? Unfortunately, there has been no work at all to determine such criteria. As a starting point, it would seem reasonable to determine how a given document has been used by people who, with existing indexing methods, managed to find it. There are then two questions we wish to ask:

1) Is there any uniformity between different users of one document in what they consider the "content" of the document?

2) If there is such uniformity, are there clues, in the text, which would help the indexer predict this user-defined "content?"

Until recently, it would have been impossible to try to answer these questions. Once indexed, a document was stored, then used at various times by different people - but no feasible method of determining who had used a particular document was available. Now, while it may still be difficult to contact the users of a document, it is no longer impractical to find some of them at least. The mechanism for doing so is the citation index. The citation index lists, for each document in it, those authors (and/or papers) who have cited it and who may be assumed to have used it.

* Words or phrases can be identified. Content, or meaning, however, involves both a word and a user.

Some of the questions that could be asked, for example, would be: "to what extent do words in the title of a document occur in the titles of documents that cited it. How has a given document been used by various authors (specific data obtained from it, experimental methods obtained, just scanned for general interest, etc.). What portion of the document was used by different users (full text, title, first and last paragraphs, abstract), how would different users have indexed the document, etc..."

Concluding Remarks

This paper has identified some costs and parameters related to indexing, interrelated these costs and parameters qualitatively, and suggested several experiments needed to make design of indexing systems a conscious and rational process. The paper is clearly neither complete nor definitive. The discursive presentation of the interractions of parameters and costs should be replaced with precise formulation. Experiments must then be undertaken to provide some insight into the quantities involved. Perhaps then the current competitive and argumentative procedures for deciding on an indexing system for a given application can be replaced with rational ones.

REFERENCES

1. Reisner, P., "Constructing an "Adaptive" Thesaurus by Man-Machine Interaction," Final Report on Applied Research Program Aerospace Intelligence Data System (AIDS) Contract AF19(626)-10, May 15, 1964, Appendix II-B.

2. Reisner, P., Towards Design and Evaluation of Indexing Systems for Information Retrieval Part I: The Indexing Problem; Quarterly Report No. 2, Computer Programming Techniques for Intelligence Analyst Application, Contract AF 30(602)-3303.

3. Lancaster, F.W. and Mills, J., "Testing Inde[illegible] and Index Language Devices" American Documentation, Ja[illegible] 1964 pp. 4-13

4. Swanson, D.R., "Searching Natural Language Text by Computer" Science, Oct.21, 1960, pp. 1099-1104

5. Kochen, M., "Toward Document Retrieval Theory: Relevance-Recall Ratios for Texts Containing One Specified Query Term" (Inter-action,"Final Report on Applied Research Program Aerospace Intelligence Data System (AIDS) Contract AF19(626)-10, May 15, 1964, Appendix III-B.

6. Cleverdon, C.W., "Report on the Testing and Analysis of an Investigation into the Comparative Efficiency of Indexing Systems." ASLIB Cranfield Research Project, Oct. 1962.

7. Arthur D. Little, Inc. "Centralization and Documentation," Final Report to the National Science Foundation, C-64469, July 1963.